JavaScript Loops

Loops in JavaScript allow you to repeat a block of code multiple times. This is useful when you want to execute the same code over and over again, like iterating through items in a list or repeating an action until a condition is met.

1. `for` Loop

The **`for` loop** is one of the most common types of loops. It repeats a block of code a specific number of times, based on a condition.

Syntax:

```
for (initialization; condition; update) {
    // code to be executed
}
```

- **Initialization**: Initializes a variable, usually a counter (e.g., `let i = 0`).
- **Condition**: The loop runs as long as this condition is true.
- **Update**: After each iteration, the counter is updated (e.g., `i++` to increase the counter).

Example:

```
for (let i = 0; i < 5; i++) {
    console.log("Iteration number: " + i);
}
```

Explanation:

- The loop starts with `i = 0`.
- It continues running as long as `i < 5`.
- After each iteration, `i` is increased by 1 (`i++`).

- Output:

```
Iteration number: 0
Iteration number: 1
Iteration number: 2
Iteration number: 3
Iteration number: 4
```

2. `while` Loop

The `while` **loop** repeats a block of code **as long as a condition is true**. It's useful when you don't know beforehand how many times you need to loop.

Syntax:

```
while (condition) {
    // code to be executed
}
```

Example:

```
let count = 0;

while (count < 3) {
    console.log("Count is: " + count);
    count++; // increase the counter
}
```

Explanation:

- The loop runs as long as `count < 3`.

- After each iteration, `count` is increased by 1.
- Output:

```
Count is: 0
Count is: 1
Count is: 2
```

3. `do...while` Loop

The **`do...while` loop** is similar to the `while` loop, but with one key difference: **it runs the code at least once**, even if the condition is false, because the condition is checked **after** the code runs.

Syntax:

```
do {
    // code to be executed
} while (condition);
```

Example:

```
let count = 0;

do {
    console.log("Count is: " + count);
    count++;
} while (count < 3);
```

Explanation:

- The loop first runs the block of code once, then checks the condition (`count < 3`).
- If the condition is true, it runs again.

- Output:

```
Count is: 0
Count is: 1
Count is: 2
```

4. `for...of` Loop

The `for...of` **loop** is used to iterate over iterable objects like arrays, strings, or sets. It loops through **values** (not just indexes).

Syntax:

```
for (variable of iterable) {
    // code to be executed
}
```

Example (with an array):

```
let fruits = ["apple", "banana", "orange"];

for (let fruit of fruits) {
    console.log(fruit);
}
```

Explanation:

- The loop goes through each value in the `fruits` array.
- Output:

```
apple
banana
orange
```

5. `for...in` Loop

The `for...in` **loop** is used to iterate over the **properties of an object**. It loops through keys (property names) in an object.

Syntax:

```
for (key in object) {
    // code to be executed
}
```

Example:

```
let person = {
    name: "John",
    age: 25,
    city: "New York"
};

for (let key in person) {
    console.log(key + ": " + person[key]);
}
```

Explanation:

- The loop goes through each property of the `person` object.
- Output:

```
    name: John
    age: 25
    city: New York
```

6. `break` and `continue` Keywords

- **`break`**: Stops the loop entirely.
- **`continue`**: Skips the current iteration and continues to the next one.

Example with `break`:

```
for (let i = 0; i < 5; i++) {
    if (i === 3) {
        break;  // stops the loop when i is 3
    }
    console.log(i);
}
// Output: 0, 1, 2
```

Example with `continue`:

```
for (let i = 0; i < 5; i++) {
    if (i === 3) {
        continue;  // skips the current iteration when i is 3
    }
    console.log(i);
}
// Output: 0, 1, 2, 4
```

Summary of Key JavaScript Loops

1. **`for` loop**: Repeats a block of code a specific number of times.
2. **`while` loop**: Repeats a block of code as long as a condition is true.
3. **`do...while` loop**: Runs the block of code at least once, then continues if the condition is true.
4. **`for...of` loop**: Loops through values of iterable objects like arrays or strings.
5. **`for...in` loop**: Loops through properties (keys) of an object.
6. **`break`** and **`continue`**: Control the flow inside loops by stopping or skipping iterations.

Example Combining Loops:

```
// Using a for loop to iterate through numbers
for (let i = 0; i < 3; i++) {
    console.log("For loop: " + i);
}

// Using a while loop to count down
let count = 3;
while (count > 0) {
    console.log("While loop count: " + count);
    count--;
}

// Using a for...of loop to iterate through an array
let colors = ["red", "green", "blue"];
for (let color of colors) {
    console.log("For...of loop: " + color);
}
```

With these loops, you can repeat actions, iterate over collections, and make your programs more powerful and dynamic!