Lecture Template

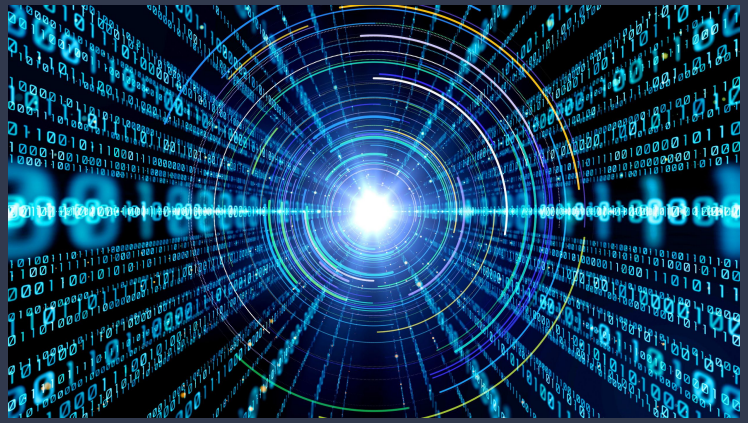
JavaScript Loops

Topics

- Introduction
  - For Loops
  - Nested Loops
  - Increment and Decrement
  - While Loops
  - Do While Loops

JavaScript Loops

Intro to Loops



In computer programming, a loop is a sequence of instruction s that is continually repeated until a certain condition is reached.

As long as the loop condition is true, the block of code associated with the condition is executed.

An infinite loop is one that lacks a functioning exit routine.

The result is that the loop repeats continually until the operating system senses it and terminates the program with an error or until some other event occurs

Loops Intro

Let's say you want to run a function, bounceBall, four times. How would you do it? L

```
function bounceBall() {   // bounce the ball here }
```

```
bounceBall() bounceBall() bounceBall() bounceBall()
```

This approach is great if you need to bounceBall only for a few times. What happens if you need to bounceBall for a hundred times?

For Loops

For Loop

The for loop runs a block of code as many times as you want to. Here's a for loop that runs bounceBall ten times:

```
for (let i = 0; i < 10; i++) {   bounceBall() }
```

It's broken down into four parts — the initialExpression, the condition, the incrementalExpression , and the statement:

```
for (initialExpression; condition; incrementExpression) {   statement }
```

1. Initialization (start)

   The initialization expression initializes the loop. The initialization expression is executed only once when the loop starts. You typically use the initialization is to initialize a counter variable.

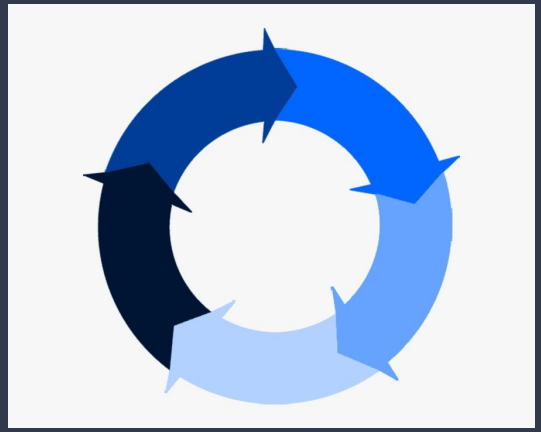
2. Condition

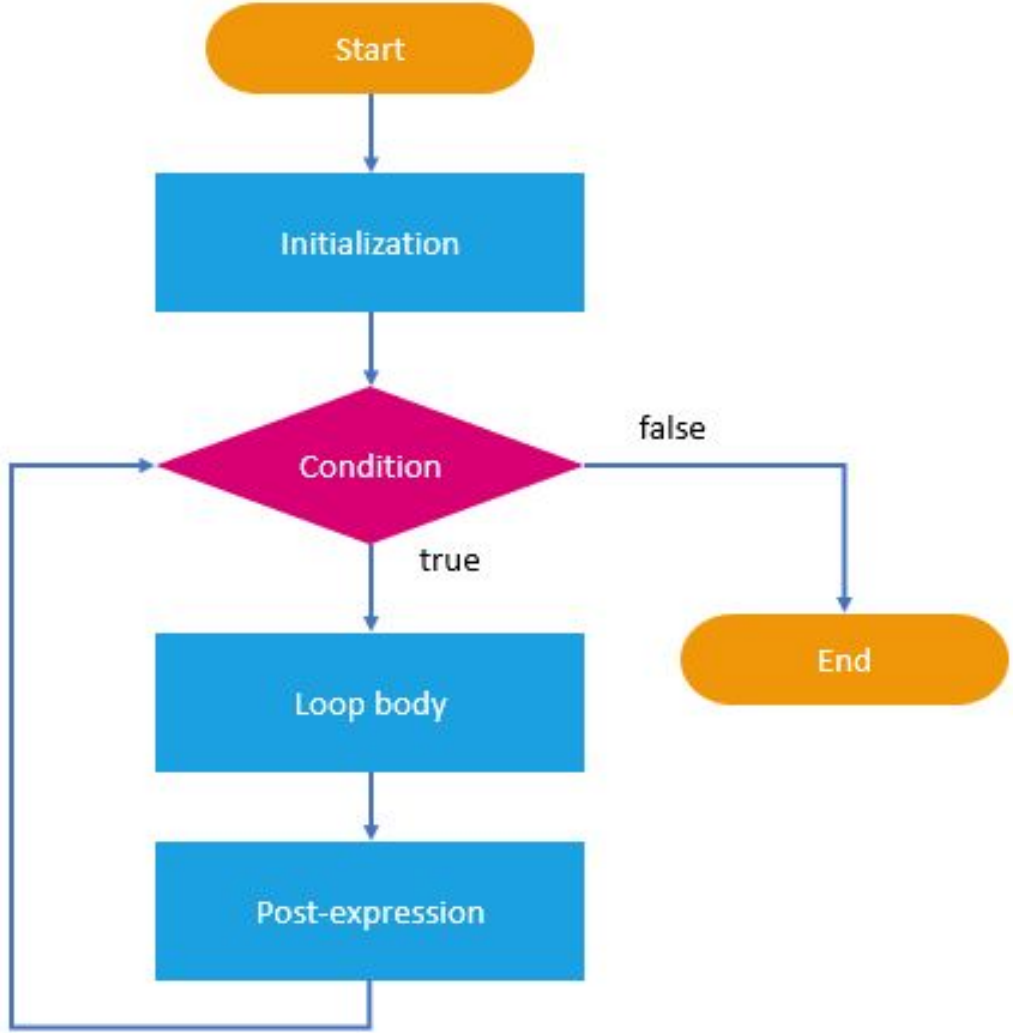
   The condition is an expression that is evaluated once before every iteration. The statement inside the loop is executed only when the condition evaluates to true. The loop is terminated if the condition evaluates to false. Note that the condition is optional. If you omit it, the for loop statement considers it as true.

3. Post (Stop)

   The for loop statement also evaluates the post-expression after each loop iteration. Generally, you use the post-expression to update the counter variable.

For Loop Flow Chart





The following example uses the for loop statement that shows the numbers from 1 to 5 in the

```
for (var counter = 1; counter < 5; counter++) {
    console.log('Inside the loop:' + counter);
}
console.log('Outside the loop:' + counter);
```

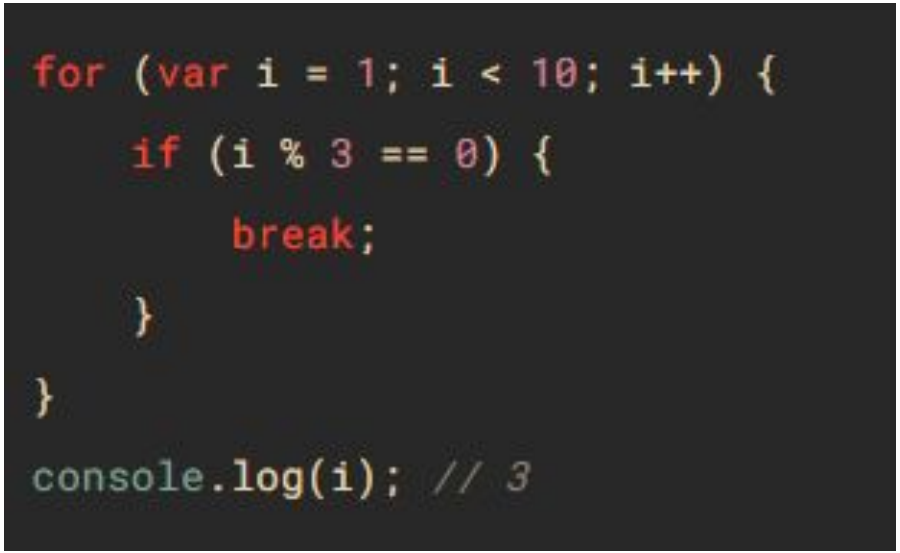
```
Inside the loop:1
Inside the loop:2
Inside the loop:3
Inside the loop:4
Outside the loop:5
```

Simple For Loop Example

Break Statements

Break Statement

The break statement gives you fine-grained control over the execution of the code in a loop. The break statement terminates the loop immediately and passes control over the next statement after the loop.



Breaking out of Loop Example

Break out of a loop before condition expression is false

```
for (var elephant = 1; elephant < 10; elephant+=2) {
    if (elephant === 7) {
        break;
    }
    console.info('elephant is ' + elephant);
}

output:
elephant is 1
elephant is 3
elephant is 5
```

All three expressions of the for loop statements are optional therefore you can omit all of them. Again, you must use a break statement to terminate the loop and also modify the counter variable to make the condition for the break statement becomes true at some point.

```
// initialize j variable
let j = 1;
for (;;) {
    // terminate the loop if j is greater than 10;
    if (j > 10) break;
    console.log(j);
    // increase the counter j
    j += 2;
}
```

For Loop without any Expressions

Infinite Loops

Infinite loops occur when the **condition** for your for loops always return true. Your browser will hang if you run an infinite loop.

To recover from an infinite loop, you need to quit your browser forcefully. On a Mac, this means you right click on your browser icon and select “force quit.” On a Window’s machine, you open the Windows Task manager with ctrl + alt + del, select your browser, and click “End task.”

Video - Loops

Nested Loops

Nested Loop

The for loop can be nested inside each other.

```
for (var x = 0; x < 5; x = x + 1) {
  for (var y = 0; y < 3; y = y + 1) {
    console.log(x + "," + y);
  }
}
```

Prints:

0, 0
0, 1
0, 2
1, 0
1, 1
1, 2
2, 0
2, 1
2, 2
3, 0

Nested Loop continued..

For each value of x in the outer loop, the inner for loop executes completely. The outer loop starts with x = 0, and then the inner loop completes its cycle with all values of y:

```
x = 0 and y = 0, 1, 2 // corresponds to (0, 0), (0, 1), and (0, 2)
```

Once the inner loop is done iterating over y, then the outer loop continues to the next value, x = 1, and the whole process begins again.

```
x = 0 and y = 0, 1, 2 // (0, 0) (0, 1) and (0, 2)
x = 1 and y = 0, 1, 2 // (1, 0) (1, 1) and (1, 2)
x = 2 and y = 0, 1, 2 // (2, 0) (2, 1) and (2, 2)
etc.
```

Increment and Decrment

Increment and Decrement

Here are some increment and decrements used in Javascript Looping.

```
x++ or ++x // same as x = x + 1
x-- or --x // same as x = x - 1
x += 3 // same as x = x + 3
x -= 6 // same as x = x - 6
x *= 2 // same as x = x * 2
x /= 5 // same as x = x / 5
```