for, while, do .... while loop

JavaScript for Loop

The for loop statement creates a loop with three optional expressions.

for (**initializer**; **condition**; **iterator**)

{

// statements

}

1) **initializer**

The for statement executes the initializer only once the loop starts. Typically, you declare and initialize a local loop **variable** in the initializer.

2) **condition**

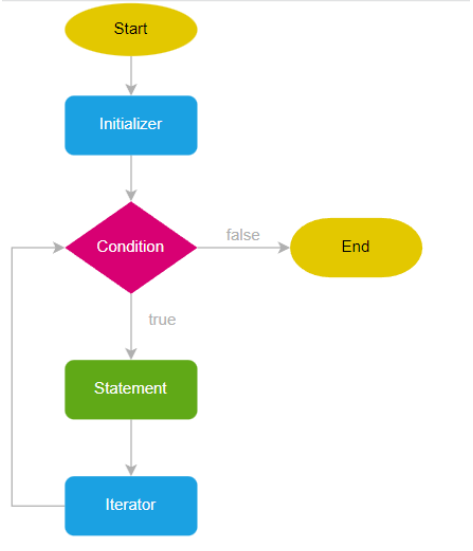
The condition is a **boolean expression** that **determines** whether the for should execute the next iteration.

The for statement evaluates the condition **before each iteration**. If the condition is true, it executes the **next iteration**. Otherwise, it'll end the loop.

3) **iterator**

The for statement executes the iterator after **each iteration.**

The following flowchart illustrates the for loop:



Example:

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

 {

  console.log(i);

}
```

How it works.

- First, declare a variable counter and initialize it to 1.
- Second, display the value of counter in the console if counter is less than 5.
- Third, increase the value of counter by one in each iteration of the loop.

In the for loop, the three expressions are optional. The following shows the for loop without any expressions:

```
// Using the JavaScript for loop without the initializer example

let j = 1;

for (; j < 10; j += 2) {

  console.log(j);

}

// Using the JavaScript for loop without the condition example

for (let j = 1; ; j += 2)

{

   console.log(j);

   if (j > 10)

   {

     break;

   }

 }

  // Using the JavaScript for loop statement without any expression example

let k = 1;

for (;;) {
```

```
 if (k > 10) {
   break;
 }
 console.log(k);
 k += 2;
}
```

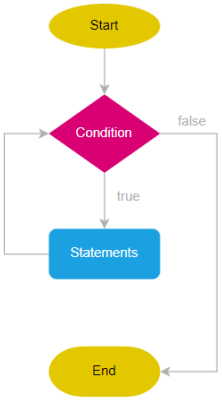
JavaScript while Loop

The JavaScript **while** statement creates a loop that executes a block as long as a condition evaluates to **true**.

```
while (expression)
 {
    // statement
   // statement
 }
```

- The while statement evaluates the expression **before each iteration** of the loop.
- If the expression **evaluates to true,** the while **statement executes** the statement. Otherwise, the while loop exits.
- Because the while loop evaluates the expression before each iteration, it is known as a **pretest loop.**
- If the expression evaluates to **false** before the **loop enters**, the while loop will never execute.

The following flowchart illustrates the while loop statement:



Example:

```
let count = 1;

while (count < 10)

 {

   console.log(count);

   count +=2;

}
```

How the script works

- First, declare and initialize the count variable to 1.
- Second, execute the statement inside the loop if the count variable is less than 10. In each iteration, output the count to the console and increase the count by 2.
- Third, after 5 iterations, the count is 11. Therefore, the condition count < 10 is false, the loop exits.

JavaScript do...while Loop

The **do...while loop** statement creates a loop that executes a block until a condition evaluates to false.

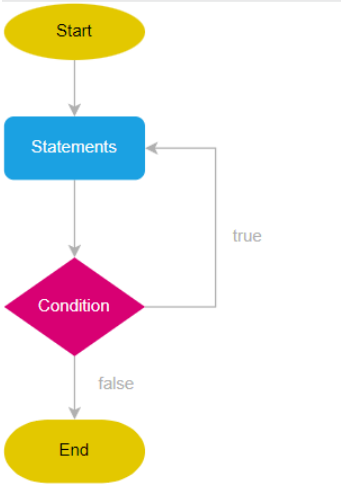
```
do {

 statement;

} while(expression);
```

- The do-while loop always executes the statement at **least once before evaluating the expression.**
- Because the do...while loop evaluates expression **after** each iteration, it's often referred to as a **post-test loop.**
- Inside the loop body, you need to make **changes to some variables** to ensure that the expression **is false** after some iterations. Otherwise, you'll have an indefinite loop.

The following flowchart illustrates the **do...while loop** statement:



Example:

```
let count = 0;

do {

 console.log(count);

 count++;

} while (count < 5)
```

How it works

- First, declare and initialize the count variable to zero.
- Second, show the count and increase its value by one in each iteration until its value is greater or equal to 5.

Project: -

1. Write a JS code to calculate the sum of digits in a number ?

   hint if num=2453 then the answer should be 14

   2+4+5+3 = 14