Loops and Jump Statements

Loops

Loops are used to do something repeatedly. You need to print 'n' numbers, then you can use a loop to do so. There are many different kinds of loops, but they almost do the same thing. Their use varies depending on the type of situation, where one loop will be easy to implement over another.
They are used to run the same code repeatedly, each time with a different value.

Types of loops

- **for**
- **for/in**
- **for/off**
- **while**
- **do-while**

for loop

A for loop is used to repeat something until the condition evaluates to false.

Syntax :

```
for ( [initializationStatement] ; [condition] ; [updateStatement] ) {
  //code to be executed multiple times
 }
```

- The *initialisation statement* is used to initialise loop counters.
- The *condition* is the expression that is evaluated to boolean value **true** or **false.**
- The *update statement* is used to update the loop counters.

Example :

```
for (let i = 0; i < 5; i++) {
   console.log("Coding Ninjas") ;
}
```

Output : Prints Coding Ninjas 5 times in 5 different lines

Initialisation statement

- The *initialisation statement* is optional, and you can initialise these statements before the **for** loop.

Example :
```
let i=0 ;
for ( ; i < 5; i++) {
   console.log("Coding Ninjas") ;
}
```
Output : Prints Coding Ninjas 5 times in 5 different lines

Note: You need to provide a semicolon for each part of the loop, even if the options are missing.

- You can provide more than one *initialisation statement* using a comma(",") as a separator.

Example :
```
for (let i = 1 , j=3 ; i <= 3 ; i++ , j--) {
   console.log( i+j ) ;
}
```
Output :
```
4
4
4
```

Note: When using multiple initialisation variables , it is not mandatory to mention conditions for each variable, but it is compulsory to mention an update statement for each of them

Condition

Often the condition is used to evaluate the condition of the initial variable.
This is not always the case; JavaScript doesn't care. Condition is also optional.
If the condition returns true, the loop will start over again. If it returns false, the loop will end.
Note: If you discard the condition, a **break** should be given inside the loop. Otherwise, the loop will never end. This will crash your browser.
We Will study about break statement further in this module

Update Statement

The update is used to increment/decrement the value of the initial variable.
This is not always the case, JavaScript doesn't care, and Update Statement is optional.
The update can also be omitted (like when you increment your values inside the loop)

Example :
```
let i = 0 ;
for ( ; i < 5; ) {
   console.log("Coding Ninjas") ;
   i++ ;
 }
```
Output : Prints Coding Ninjas 5 times in 5 different lines

Loop scope

Using var :

```
var i = 5;
for (var i = 0; i < 10; i++) {
  console.log("Coding Ninjas") ;
}
console.log(i) ;  // 10
```
Because till 9, the condition would be true, then i increments by one and becomes 10 and exits the loop.

By using var, the variable declared in the loop redeclares the variable outside the loop.

Using let:

```
let i = 5;
for (let i = 0; i < 10; i++) {
  console.log("Coding Ninjas") ;
}
console.log(i) ;  // 5
```

By using let, the variable declared in the loop is not redeclared outside the loop, and **i** used in the loop**(block scope)** is different from the outside variable **i**

For/in and For/of Loops

- **for/in** - loops through the properties of an object
- **for/of** - loops through the values of an iterable object

They are mainly. used on objects and arrays, hence will be covered in detail in the next module

while loop

The while statement executes the statements until the condition is not **false**.

Syntax :
```
while (condition) {
  // code to be executed multiple times
}
```

First, the condition is evaluated, and if it is true, then the statements are executed, and the condition is tested again. The execution stops when the condition returns false.

NOTE: You have to provide an **update** expression inside the loop so that it does not repeat infinitely.

Example :
```
let i = 0 ;
while (i < 5) {
  console.log("Coding Ninjas") ;
  i++;
 }
```
Output : Prints Coding Ninjas 5 times in 5 different lines

do while loop

The do-while loop is similar to the while loop, except that the statements are executed at least once

Syntax :
```
do {
    // code to be executed multiple times
} while (condition);
```

Example :
```
let i = -1 ;
do {
  console.log("Coding Ninjas") ;
  i++;
} while (i > 0);
```
Output : Coding Ninjas

Difference between while and do while loop

while	do while
Condition is checked first then statement(s) is executed.	Statement(s) is executed at least once. After that, the condition is checked.
It might occur statement(s) is executed zero times If the condition is false.	At least once the statement(s) is executed.
No semicolon at the end of a while. while(condition)	The semicolon at the end of the while. while(condition);
If there is a single statement, brackets are not required.	Brackets are always required.
Variable in the condition is initialised before the execution of the loop.	Variable may be initialised before or within the loop.
while loop is entry controlled loop.	do while loop is exit controlled loop.

Jump Statements

In JavaScript there are two types of jump statements :

- break
- continue

break Statement

We talked about this in the for loop when there is no condition passed then it is used to **jump out** of a switch( ) statement.
The break statement can also be used to jump out of a loop.

```
Example :  for(let i = 0 ; ; i++){
               if(i==3)
                  break ;
               console.log("Coding Ninjas") ;
           }
```
Output : Prints Coding Ninjas 3 times in 3 different lines

NOTE: In the above example, there was no condition passed to the for loop, but in case the condition is mentioned, the break can also be used to jump out of the loop at some point

```
Example :  for(let i = 0 ; i < 10 ; i++){
               if(i==3)
                  break ;
               console.log("Coding Ninjas") ;
           }
```
Output : Prints Coding Ninjas 3 times in 3 different lines

continue Statement

The **continue** statement breaks one iteration (in the loop) if a specified condition occurs and continues with the next iteration.

For example, you need to print "Coding Ninjas" if the number is odd and the number along with it from 0 to 10

Example :

```
for (let i = 0; i < 10; i++) {
  if ( i %2 == 0)
      continue ;
 console.log("Coding Ninjas" , i ) ;
 }
```

Output :
Coding Ninjas 1
Coding Ninjas 3
Coding Ninjas 5
Coding Ninjas 7
Coding Ninjas 9

This could've also be achieved by using **if-else.**