Chapter 3 - Loops & functions

We use loops to perform repeated actions. For example - If you are assigned a task of printing numbers from 1 to 100, it will be very hectic to do it manually, Loops help us automate such tasks

Types of loops in JavaScript

for loop → loop a block of code no of times
for in loop → loops through the keys of an Object
for of loop → loops through the values of an object
while loop → loops a block based on a specific condition
do-while loop → while loop variant which runs atleast once

The for loop

The syntax of a for loop looks something like this

```
for ( statement 1 ; statement 2; statement 3) {
  // code to be executed
}
```

→ Statement 1 is executed one time
→ Statement 2 is the condition base on which the loop runs (loop body is executed)
→ Statement 3 is executed everytime the loop body is executed

Quick Quiz : Write a sample for loop of your choice.

The for-in loop
The syntax of for-in loop looks like this

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

Quick Quiz: Write a sample program demonstrating for-in loop

Note - for-in loops also work with arrays which will be discussed in the later videos.

The for-of loop
The syntax of for-of loop looks like this

```
for (variable of iterable) {
  // code
}
```

↳ For every iteration
↳ Iterable data structure like Arrays, Strings etc.

Quick Quiz: Write a sample program demonstrating for-of loops

The while loop
The syntax of while loop looks like this:

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

Note: If the condition never becomes false, the loop will never end and this might crash the runtime!

Quick Quiz: Write a sample program demonstrating while loop.

The do-while loop

The do while loop's syntax looks like this:

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

=> Executed at least once

Quick Quiz: Write a sample program demonstrating do while loop

Functions in Java Script

A JavaScript function is a block of code designed to perform a particular task.

Syntax of a function looks something like this:

```
function myFunc () {
    // code
}
```

```
function bindFunc (parameter 1, parameter 2) {
    // code
}
```

↳ Function with parameters

Here the parameters behave as local variables

binod func (7, 8) ⟹ Function Invocation

Function invocation is a way to use the code inside the function

A function can also return a value. The value is "returned" back to the caller

```
Const sum = (a, b) => {
                let c = a + b;
                return c;
                }
```

↓ Another way to create & use the function ↑

⟹ Returns the sum

```
let y = sum (1, 3)
console.log(y)
```

→ Prints 4