```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        /* CSCI-111 Web Programming and Problem Solving */
        const instructors = ["Dr. Talgat Manglayev", "Dr. Irina Dolzhikova", "MSc. Marat Isteleyev"]
        while(true)
        {
            console.log("week-9-lecture")
            console.log("Loop")
        }
    </script>
</body>
</html>
```

Content

- Introduction
- For loop
- While loop

Introduction

Problem: We need to compute the average grade for lab:

$$\texttt{let lab\_1\_average} = \frac{\texttt{(student\_1.lab\_1 + student\_2.lab\_1 + student\_3.lab\_1 + ... + student\_39.lab\_1)}}{\texttt{n}}$$

What if you have 500 students in your class?

Introduction

To perform some **repetitive** tasks we use **loops.**

Loops are JavaScript constructs that allow us to perform the repetitive tasks:

- *for* a specified number of times;

- *while* a specified condition holds true.

For Loop

Syntax:

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

where:

- ***Expression 1*** *is executed (one time) before the execution of the block code.*
- ***Expression 2*** *defines the condition for executing the code block.*
- ***Expression 3*** *is executed (every time) after the code block has been executed.*

For Loop

```
let lab_1 = [4, 5 ,0 ,4 ,5 ,10 ,10]

let sum = 0;

for (let i = 0; i < lab_1.length; i++)

{

    sum = sum + lab_1[i];

}
console.log("sum = "+ sum)
```
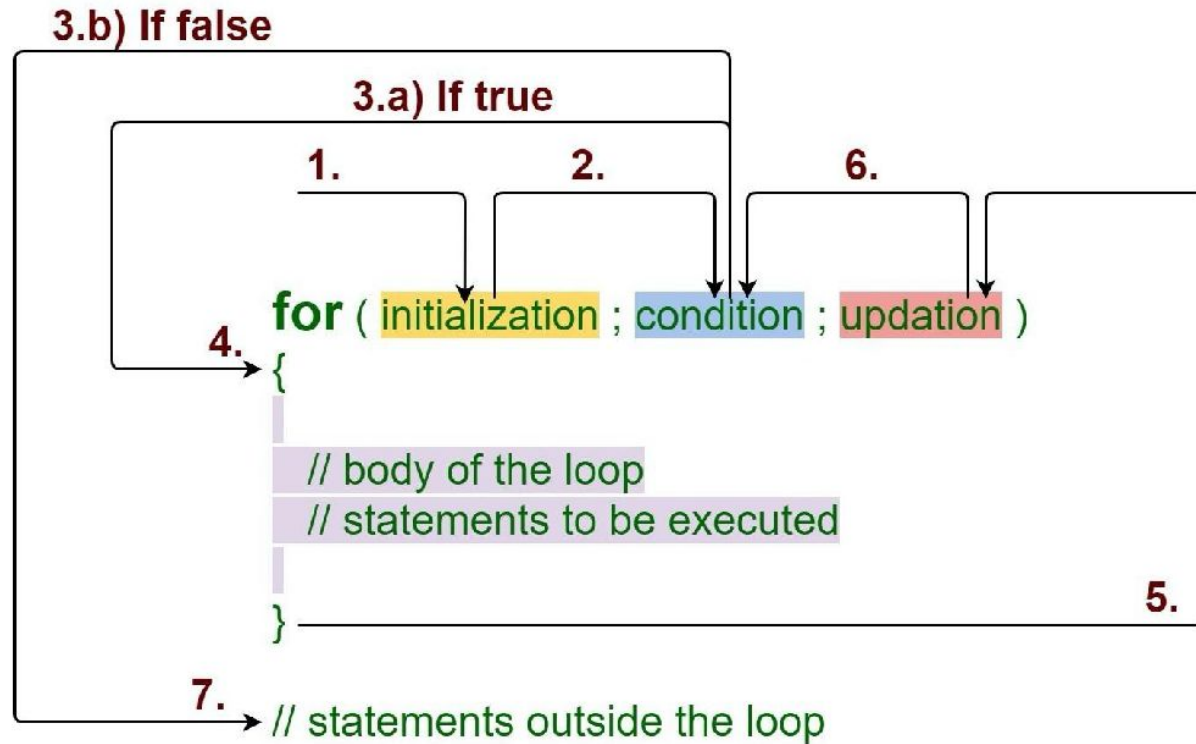
i - loop iterable

Three steps of the loop:

1) `let i = 0` - initialization

2) `i < lab1_1.length` - stop condition

3) `i++` - update rule

For Loop

For Loop

3.b) If false

3.a) If true

1.   2.   6.

```
for ( initialization ; condition ; updation )
4.
{

    // body of the loop
    // statements to be executed

}                                        5.

7.
// statements outside the loop
```

GeeksforGeeks

Nested For Loop

```javascript
for (let i = 0; i < 5; i++)

{

  for (let j = 0; j < 10; j++)

  {

    console.log("Hello, World!")

  }

}
```

While Loop

Syntax:

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

where:

- **Condition** *is a logical expression for executing the code block*
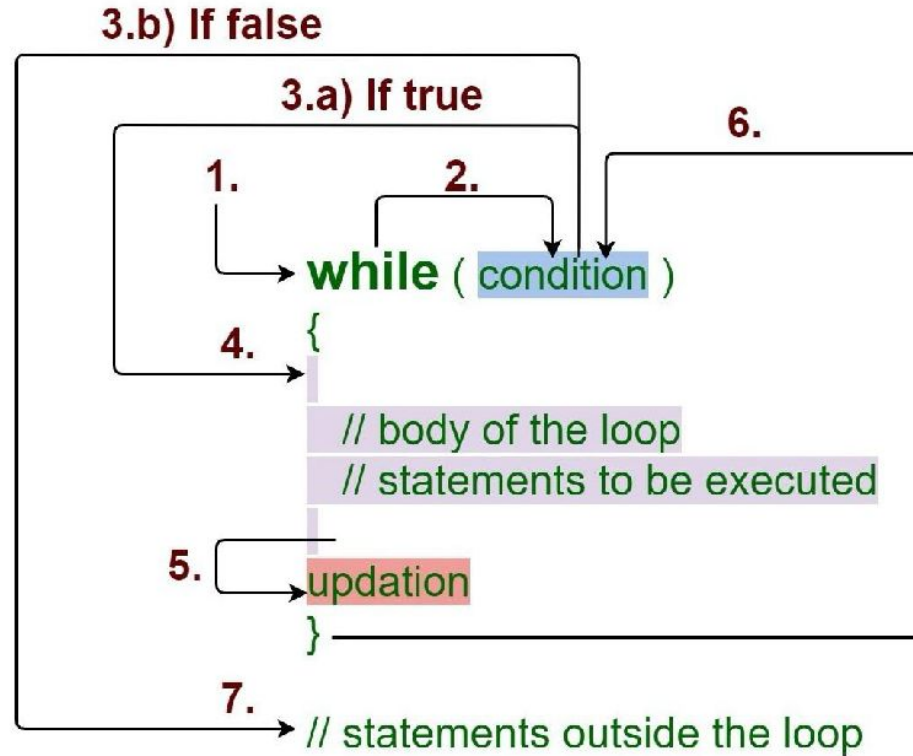
While Loop

```
let lab_1 = [4, 5 ,0 ,4 ,5 ,10 ,10];

let sum = 0;

let i = 0;                    // initialization of loop variable

while (i < lab_1.length)// stop condition

{

    sum += lab_1[i];       // body of the loop

    i++;                      // update the loop variable

}

console.log("sum = "+ sum)
```

There are also three steps of the loop:
1) `let i = 0` - initialization
2) `i < lab_1.length` - stop condition
3) `i++` - update rule

While Loop

While Loop

```
let i = 0;
while (i < 100)
{
    console.log("Hello, World!")     What does this loop do?
    i++;
}
```

While Loop

```
let i = 0;
while (i < 100)
{
    console.log("Hello, World!")
    i++;
}


for (let i = 0; i < 100; i++)

{

    console.log("Hello, World!")

}
```

What does this loop do?

Compare

For vs. While Loop

- Use *for*-loop when you know in advance the number of steps to do

- Otherwise use *while*-loop

Break and Continue

– The ***break*** statement exits the loop.

– The ***continue*** statement skips one iteration in the loop

```
for (let i = 0; i < 10; i++)
{
    if (i == 4)
    {
        continue;
    }
    if (i == 8)
    {
        break;
    }
    console.log(i + " Hello, World!")
}
```

Break and Continue

– The *break* statement exits the loop.

– The *continue* statement skips one iteration in the loop

```
for (let i = 0; i < 10; i++)
{
    if (i == 4)
    {
        continue;
    }
    if (i == 8)
    {
        break;
    }
    console.log(i + " Hello, World!")
}
```

```
0 Hello, World!
1 Hello, World!
2 Hello, World!
3 Hello, World!
5 Hello, World!
6 Hello, World!
7 Hello, World!
>
```

Summary

- To perform repetitive tasks, use loops. There are two types of loops:
    - ***for*** a specified number of times.
    - ***while*** a specified conditions holds.

- Don't forget about three steps of the loops:
    - Initialize the loop variables before the loop.
    - Setup the condition to exit the loop.
    - Update the value of the loop variables.

- Use ***break*** and ***continue*** commands if needed.