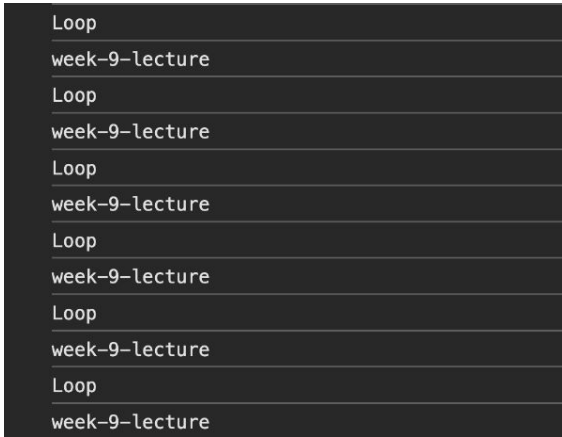
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        /* CSCI-111 Web Programming and Problem Solving */
        const instructors = ["Talgat Manglayev",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



Content

- Introduction
- For loop
- While loop
- Break and Continue statements

Introduction

Problem: Calculate the average grade for lab 1

Introduction

Problem: Calculate the average grade for lab 1

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

Introduction

Problem: Calculate the average grade for lab 1

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

What if there are 500 students in 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.*

1.html, 2.html

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:

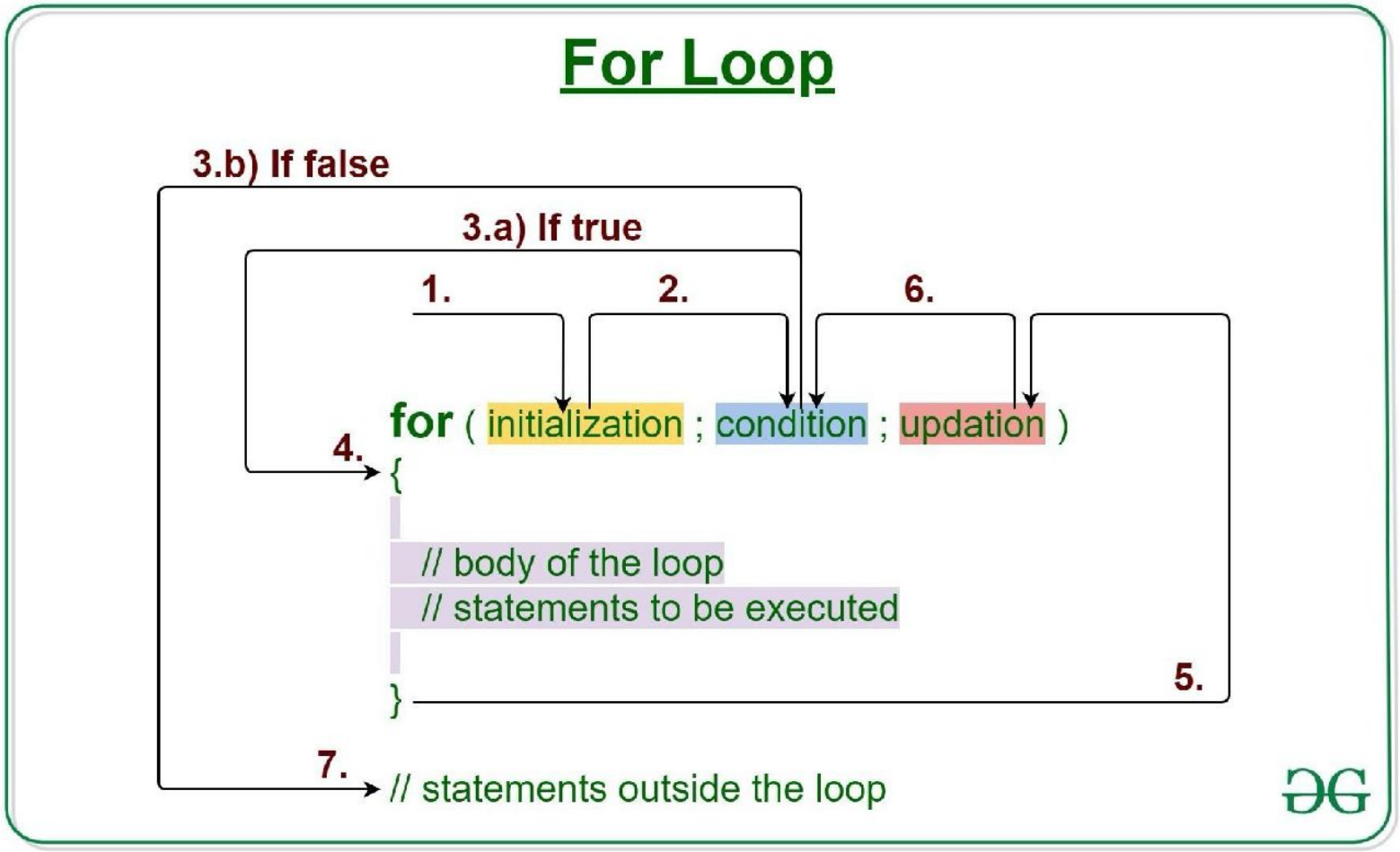
1) **`let i = 0`** - initialization

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

3) **`i++`** - update rule

3.html, 4.html

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

Nested For Loop

```
for (let i = 0; i < 5; i++)
{
  for (let j = 0; j < 10; j++)
  {
    console.log("Hello, World!")
  }
}
```

5.html

While Loop

Syntax:

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

where:

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

6.html, 7.html

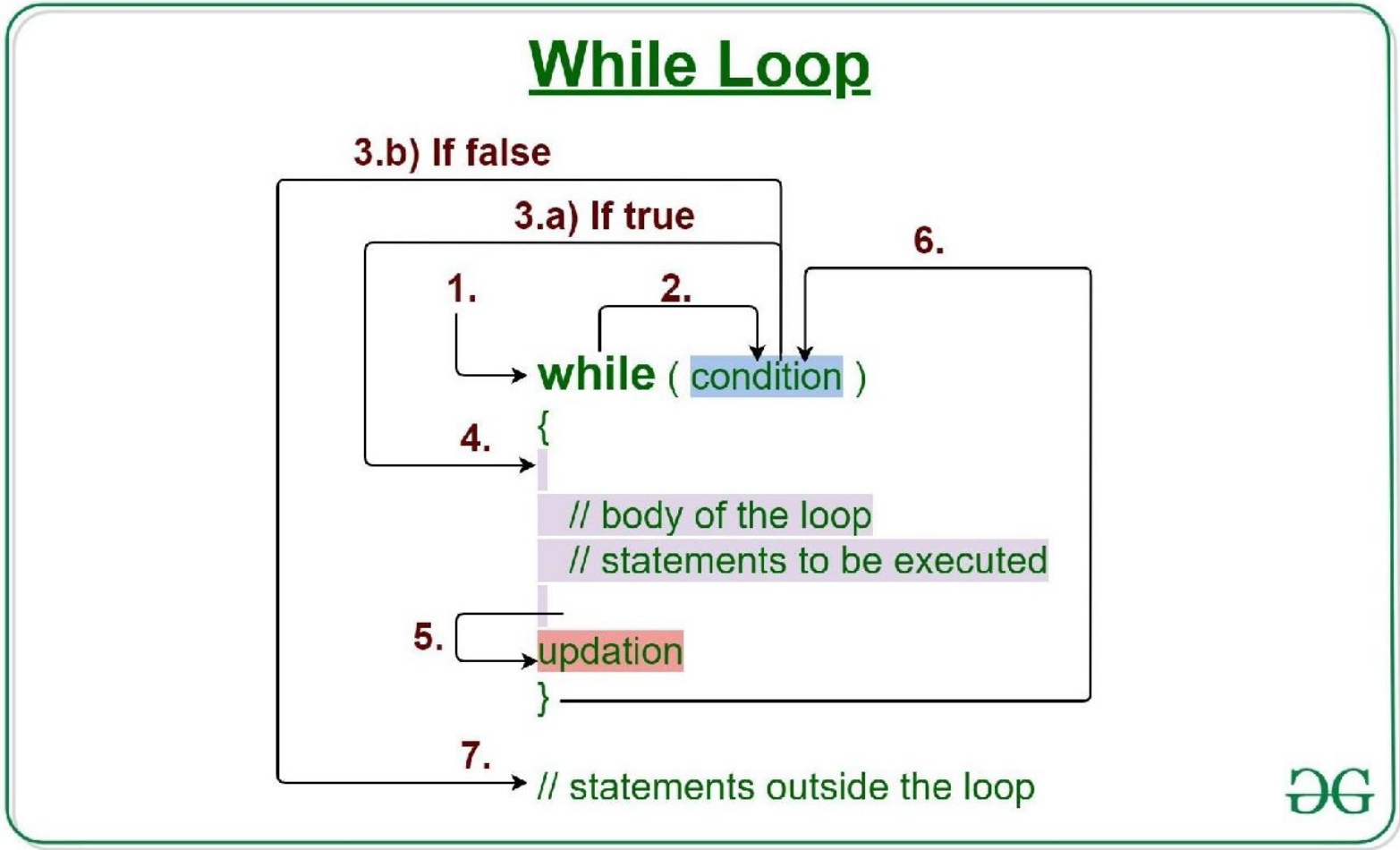
While Loop

```
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)
```

8.html, 9.html

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
3.b) If false
3.a) If true
6.
1.
2.
while (condition)
4.
{
// body of the loop
// statements to be executed
5.
updation
}
7.
// statements outside the loop

While Loop

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

What does this loop do?

While vs. For Loop

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

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

Compare
What do these loops do?

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

10.html

– 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!
>
```

Break and Continue

10.html

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

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

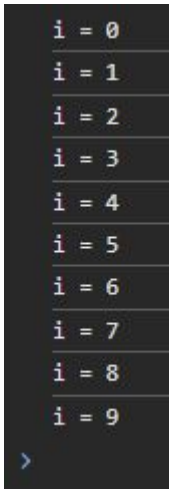
```
for (let i = 0; i < 10; i++)
{
    if (i == 4)
    {
        continue;
    }
    if (i == 8)
    {
        break;
    }
    console.log(i + " Hello, World!")
}
```

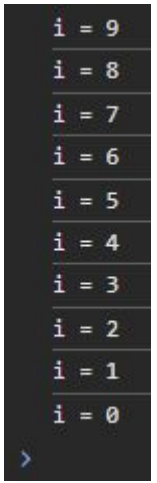
LET US REWRITE THIS EXAMPLE WITH **WHILE** LOOP!

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

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

Change the code in the left to print the output below (reverse order)





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.