Internship Report – Frontend Dev

Week 4: JavaScript Basics

Name: Zainab
Father Name: Assad Qayyum
Date: 16th July, 2025
Internship Domain: Front-end Intern
Task: JS - Loops (for, while, do-while), Arrays

Task Overview: (Day3)

Today's task was to learn about **Loops** (for, while, do-while) and **Arrays**

in JavaScript. The goal was to understand how to repeat code using loops and how to store multiple values using arrays.

Content Covered:

- Loops: for, while, and do-while
- Arrays: creation, indexing, looping through, basic operations
- Practical understanding through mini practice examples and hands-on coding

1. Loops in JavaScript:

Loops are used when we want to **run the same block of code multiple times**. There are three primary types of loops:

a) for Loop:

The for loop is useful when we know exactly **how many times** we want to run a loop.

Syntax:

```
for (initialization; condition; increment) {
  // code block to execute
}
```

- Initialization: Starts the loop (e.g., let i = 0)
- Condition: Loop continues as long as this is true (i < 5)
- Increment: Updates the counter (i++)

Example:

```
for (let i = 1; i <= 5; i++) {
  console.log("Step " + i);
}
```

b) while Loop :

The while loop is used when the number of iterations is not known in advance. It runs as long as the **condition is true.** It checks the condition before executing the block.

Syntax:

```
while (condition) {
  // code block to execute
}
```

Example:

```
let count = 1;
while (count <= 3) {
  console.log("Count: " + count);
  count++;
}
```

If the condition never becomes false, the loop runs forever (infinite loop).

c) do while loop:

This loop will **always execute once**, then continue as long as the condition is true.

Syntax:

```
do {
  // code block to execute
} while (condition);
```

Example:

```
let i = 0;
do {
  console.log("i is " + i);
  i++;
} while (i < 3);
```

This guarantees the **code runs at least once,** even if the condition is false initially.

2. Arrays in JavaScript:

An array is a special variable that can **store multiple values**. Instead of using separate variables, you can store a list of items in one array.

Key Properties:

- Indexed: Starts from 0
- Can hold any data type: strings, numbers, even other arrays
- Supports many useful methods

Declaring Arrays

You can declare an array like this:

```
Let  fruits = ["Apple", "Banana", "Mango"];
```

Accessing Values

For accessing values from an array:

```
console.log(fruits[0]);       // Apple
```

Modifying Arrays

You can modify an array like this:

```
fruits[1] = "Orange";     // Changes "Banana" to "Orange"
```

Array Methods:

Method	Description
push()	Adds item at the end
pop()	Removes item from the end
shift()	Removes item from the start
unshift()	Adds item at the start
length	Returns total number of elements

Looping Through Arrays:

With for loop:

```
for (let i = 0; i < fruits.length; i++) {
  console.log(fruits[i]);
}
```

With for…of loop:

```
for (let fruit of fruits) {
  console.log(fruit);
}
```

Practice Code:

This code displays a basic explanation of today's concepts. It demonstrates the use of JavaScript **loops (for, while, and do-while)** and **arrays** in a simple and clear way. It prints a series of numbers using each loop type and then loops through an array of fruits to display them. The output is styled and displayed inside a styled card on the webpage.

practice.html | loops-arrays.html

loops-arrays.html > html

```
<!DOCTYPE html>
<html>
<head>
  <title>Loops and Arrays Practice</title>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background-color: #eaf4fb;
      color: #333;
      padding: 40px;
      max-width: 800px;
      margin: auto;
    }

    h1 {
      text-align: center;
      color: #2c3e50;
    }

    .section {
      background: #fff;
      padding: 20px;
      border: 2px solid #2c3e50;
      border-radius: 10px;
      margin-bottom: 25px;
      box-shadow: 0 4px 10px rgba(0, 0, 0, 0.05);
    }

    #output {
      background: #eef3f7;
```

Ln 2, Col 7 Spaces: 4 UTF-8 CRLF {} HTML Port : 5500

practice.html | loops-arrays.html

loops-arrays.html > html

```
<html>
<head>
  <style>

    #output {
      background: #eef3f7;
      border-left: 4px solid #3498db;
      padding: 10px;
      border-radius: 8px;
      font-family: monospace;
      white-space: pre-line;
    }
  </style>
</head>
<body>

  <h1>Loops and Arrays in JavaScript</h1>


  <div class="section">
    <strong>Loops</strong> let you repeat actions. Common types:
    <ul>
      <li><b>for</b> – run a known number of times</li>
      <li><b>while</b> – run while a condition is true</li>
      <li><b>do-while</b> – runs at least once, then checks</li>
    </ul>
  </div>

  <div class="section">
    <strong>Arrays</strong> store multiple values in one variable. You can loop through them and use methods like:
```

Ln 2, Col 7 Spaces: 4 UTF-8 CRLF {} HTML Port : 5500

practice.html | loops-arrays.html

loops-arrays.html > html

```
2   <html>
39  <body>
53    <div class="section">
54      <strong>Arrays</strong> store multiple values in one variable. You can loop through them and use methods like:
55      <code>push()</code>, <code>pop()</code>, and <code>length</code>.
56    </div>
57
58    <div class="section">
59      <h3>Output:</h3>
60      <p id="output"></p>
61    </div>
62
63    <script>
64      let message = "";
65
66      // FOR LOOP
67      message += "FOR loop (1 to 5):\n";
68      for (let i = 1; i <= 5; i++) {
69        message += i + " ";
70      }
71      message += "\n\n";
72
73      // WHILE LOOP
74      message += "WHILE loop (Even 2 to 10):\n";
75      let j = 2;
76      while (j <= 10) {
77        message += j + " ";
78        j += 2;
79      }
80      message += "\n\n";
```

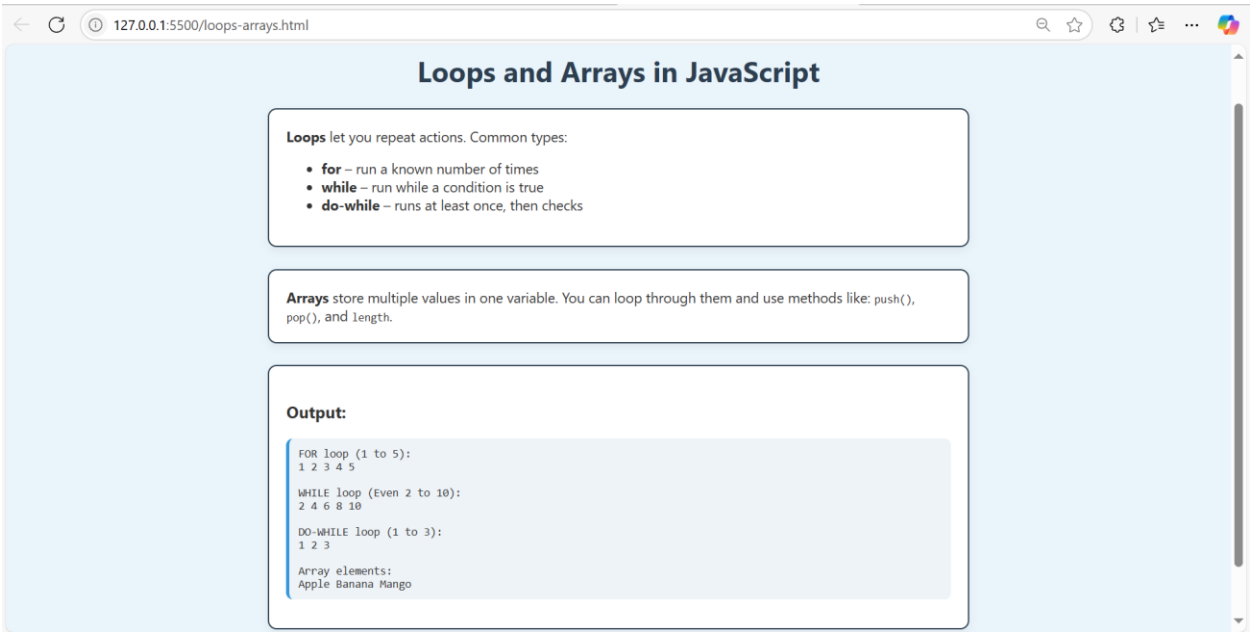
Ln 2, Col 7 Spaces: 4 UTF-8 CRLF {} HTML Port : 5500

practice.html | loops-arrays.html

loops-arrays.html > html > body > script

```
2   <html>
39  <body>
63    <script>
76      while (j <= 10) {
77        message += j + " ";
78        j += 2;
79      }
80      message += "\n\n";
81
82      // DO-WHILE LOOP
83      message += "DO-WHILE loop (1 to 3):\n";
84      let k = 1;
85      do {
86        message += k + " ";
87        k++;
88      } while (k <= 3);
89      message += "\n\n";
90
91      // ARRAY
92      let fruits = ["Apple", "Banana", "Mango"];
93      message += "Array elements:\n";
94      for (let i = 0; i < fruits.length; i++) {
95        message += fruits[i] + " ";
96      }
97
98      document.getElementById("output").innerText = message;
99    </script>
100 </body>
101 </html>
102
```

Ln 88, Col 22 Spaces: 4 UTF-8 CRLF {} HTML Port : 5500



Conclusion:

Today's lesson focused on the fundamentals of loops and arrays in JavaScript — both of which are essential for writing efficient and scalable code. Loops help reduce repetition, while arrays make handling multiple data items easier.