# INTERNSHIP REPORT

# WEEK 5 DAY 3

| | | | |
|---|---|---|---|
| Submitted to: | **Ali Hyder** | Submission Date: | **23rd July, 2025** |
| Internship Domain: | **Front Development** | Internship Name: | **ProSensia** |
| Student Name: | **Yasal Qamar** | Roll No. | **S25031** |

## JavaScript Advanced

## Topics: JavaScript Array Methods (map, filter, reduce)

### Objective

To understand how to use JavaScript array methods map(), filter(), and reduce() to manipulate and transform data in arrays effectively.

### Topics Covered

1. **.map():**
   - Used to transform each element in an array and return a new array.
   - Practiced modifying array values (e.g., multiplying numbers, transforming objects).

**Example:**

```
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(num => num * 2);
console.log(doubled); // [2, 4, 6, 8]
```

2. **.filter():**
   - Used to select elements that match a specific condition.
   - Practiced extracting even numbers, filtering objects based on conditions.

**Example:**

```
const numbers = [1, 2, 3, 4, 5];
const evens = numbers.filter(num => num % 2 === 0);
```

```
console.log(evens); // [2, 4]
```

### 3. .reduce():

- Used to reduce an array to a single value (like sum or object aggregation).
- Practiced summing array numbers, calculating totals from object arrays.

**Example:**

```
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
console.log(sum); // 10
```

## CODING

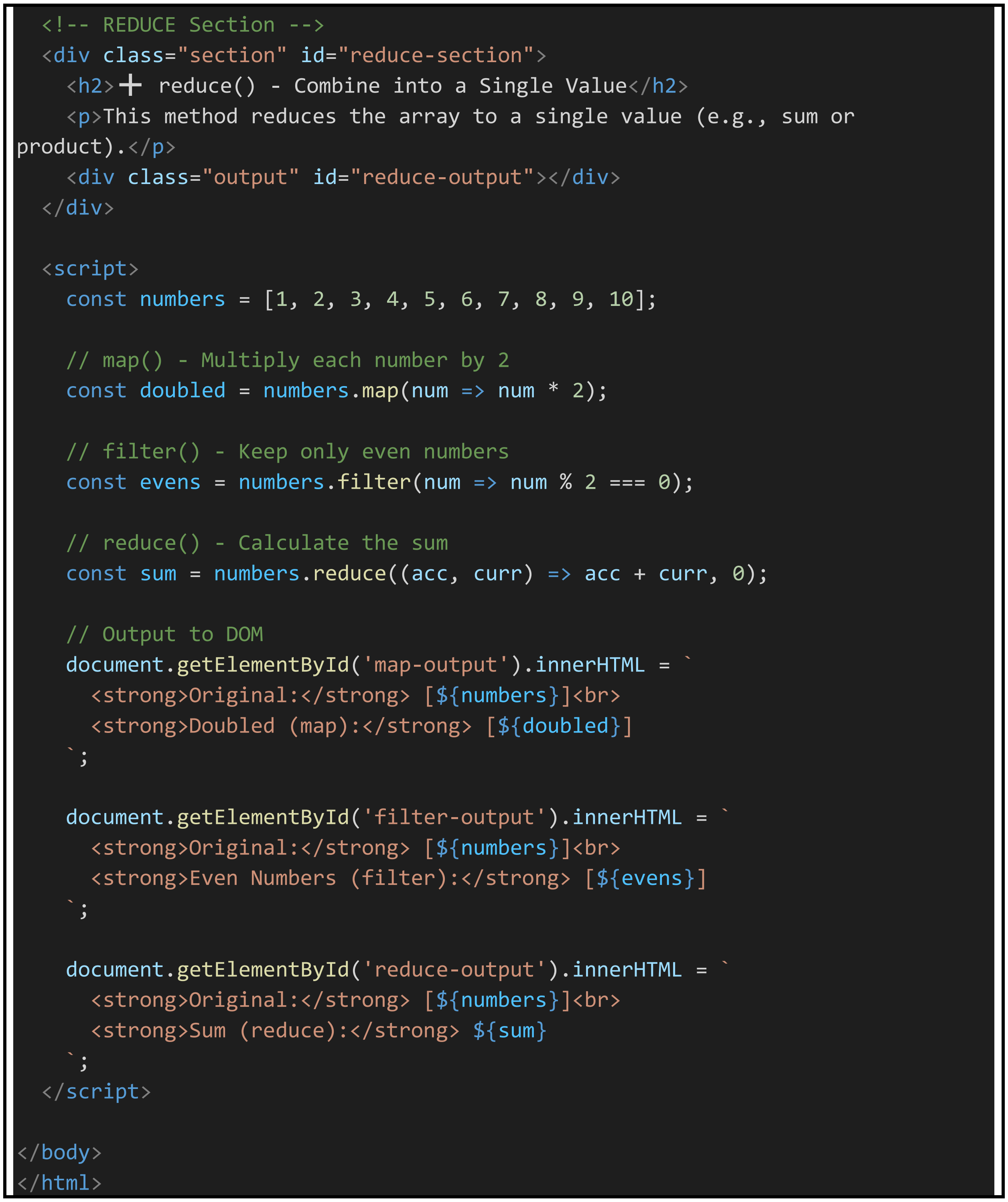
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Array Methods: map, filter, reduce</title>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background-color: #f3f4f6;
      margin: 0;
      padding: 20px;
    }
    h1 {
      text-align: center;
      color: #333;
    }
    .section {
      background-color: #fff;
      margin: 20px auto;
      padding: 20px;
      max-width: 800px;
      border-radius: 12px;
      box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    }
    h2 {
      color: #007acc;
      margin-bottom: 10px;
    }
    p {
      font-size: 16px;
      color: #444;
    }
    .output {
      background-color: #f9fafb;
      padding: 15px;
      margin-top: 10px;
      border-left: 4px solid #007acc;
      font-family: monospace;
    }
    code {
      background-color: #eef;
      padding: 2px 6px;
      border-radius: 4px;
    }
  </style>
</head>
<body>

  <h1>JavaScript Array Methods: <code>map()</code>, <code>filter()</code>,
<code>reduce()</code></h1>

  <!-- MAP Section -->
  <div class="section" id="map-section">
    <h2>🔄 map() - Transform Each Item</h2>
    <p>This method creates a new array by applying a function to each
element.</p>
    <div class="output" id="map-output"></div>
  </div>



  <!-- FILTER Section -->
  <div class="section" id="filter-section">
    <h2>✅ filter() - Keep Matching Items</h2>
    <p>This method creates a new array with only elements that match a
condition.</p>
    <div class="output" id="filter-output"></div>
  </div>
```



```
  <!-- REDUCE Section -->
  <div class="section" id="reduce-section">
    <h2>➕ reduce() - Combine into a Single Value</h2>
    <p>This method reduces the array to a single value (e.g., sum or 
product).</p>
    <div class="output" id="reduce-output"></div>
  </div>

  <script>
    const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    // map() - Multiply each number by 2
    const doubled = numbers.map(num => num * 2);

    // filter() - Keep only even numbers
    const evens = numbers.filter(num => num % 2 === 0);

    // reduce() - Calculate the sum
    const sum = numbers.reduce((acc, curr) => acc + curr, 0);

    // Output to DOM
    document.getElementById('map-output').innerHTML = `
      <strong>Original:</strong> [${numbers}]<br>
      <strong>Doubled (map):</strong> [${doubled}]
    `;

    document.getElementById('filter-output').innerHTML = `
      <strong>Original:</strong> [${numbers}]<br>
      <strong>Even Numbers (filter):</strong> [${evens}]
    `;

    document.getElementById('reduce-output').innerHTML = `
      <strong>Original:</strong> [${numbers}]<br>
      <strong>Sum (reduce):</strong> ${sum}
    `;
  </script>

</body>
</html>
```

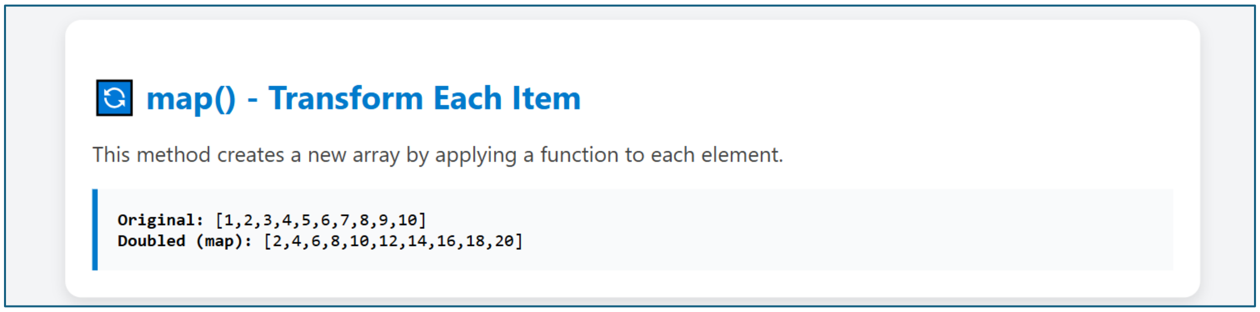
**Explanation:**

### 1. map() – Transform Each Item

```
const doubled = numbers.map(num => num * 2);
```

- numbers is an array:
  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- .map() goes through each element (num) and returns num * 2.
- A new array is created with all values doubled:

**Output:**



#### 🔄 map() - Transform Each Item

This method creates a new array by applying a function to each element.

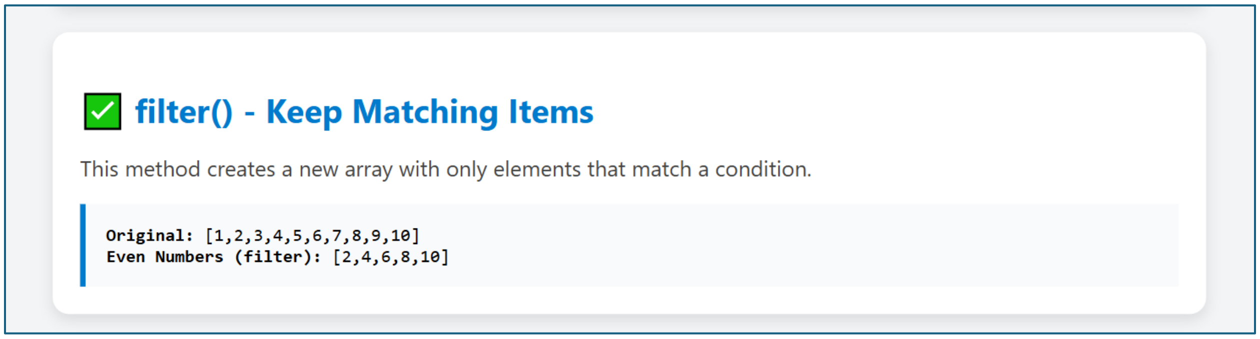
```
Original: [1,2,3,4,5,6,7,8,9,10]
Doubled (map): [2,4,6,8,10,12,14,16,18,20]
```

### 2. filter() – Keep Matching Items

```
const evens = numbers.filter(num => num % 2 === 0);
```

- It loops through each number in the numbers array.
- The condition num % 2 === 0 checks for even numbers.
- Only values that are even are kept in the new array:

**Output:**



#### ✅ filter() - Keep Matching Items

This method creates a new array with only elements that match a condition.

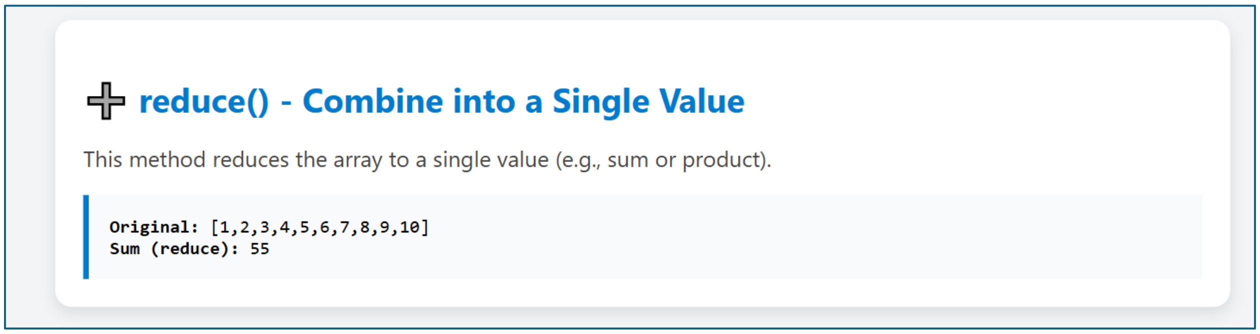
```
Original: [1,2,3,4,5,6,7,8,9,10]
Even Numbers (filter): [2,4,6,8,10]
```

### 3. reduce() – Combine into a Single Value

```
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
```

- acc = accumulator (stores the total)
- curr = current value in the array
- Starts from 0, then adds each number one by one:

**Output:**



### ➕ reduce() - Combine into a Single Value

This method reduces the array to a single value (e.g., sum or product).

```
Original: [1,2,3,4,5,6,7,8,9,10]
Sum (reduce): 55
```

| Method | What it Does | Returns | Example Output |
|---|---|---|---|
| map() | Transforms each item | New array | [2, 4, 6, ...] |
| filter() | Keeps only items that match a condition | New array | [2, 4, 6, 8, 10] |
| reduce() | Combines all values into one | Single value | 55 |

## CONCUSLION:

In this session, I gained practical experience with three powerful JavaScript array methods:

1. **map()** – helped me create a new array by transforming each element. I used it to double the values in a number array, which taught me how to modify data without changing the original.
2. **filter()** – allowed me to extract specific elements from an array based on a condition. I used it to filter out even numbers, enhancing my understanding of how to work with array conditions and logic.
3. **reduce()** – enabled me to combine all values in an array into a single result. I practiced summing all the numbers, which demonstrated how to aggregate data efficiently.

By combining these methods in one project and displaying the results on a styled webpage, I improved not only my JavaScript logic but also my ability to structure code cleanly. These techniques are essential for data processing in real-world web development.