# PROSENSIA PVT LTD

# M FAROOQ SAJID

# ARRAY METHODS

## 1: Introduction

JavaScript arrays are versatile structures that allow storing multiple values. To work efficiently with these arrays, JavaScript offers several methods. Among them, map(), filter(), and reduce() stand out for their power and flexibility in functional programming.

---

## 2: map() Method

**Purpose**: Transforms each element in the array and returns a new array.

**Example**:

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

---

## 3: filter() Method

**Purpose**: Filters elements based on a condition and returns a new array.

**Example**:

javascript
CopyEdit
```
  const numbers = [1, 2, 3, 4, 5, 6];const evenNumbers = numbers.filter(num =>
num % 2 === 0);console.log(evenNumbers); // [2, 4, 6]
```

---

## 4: reduce() Method

**Purpose**: Reduces an array to a single value using a callback function.

**Example**:

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

---

## 5: Summary and Best Practices

Use map() for transformations.

Use filter() to select a subset.

Use reduce() to summarize or aggregate data.

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

---