JavaScript Methods for Arrays of Objects

1. map

Purpose

map is used to transform each element of an array and return a new array. The original array remains unchanged.

Syntax

```
array.map(callback(currentValue, index, array))
```

Example: Add a New Property to Employees

We have a list of employees, and we need to add a tax property based on their salary (10% of the salary).

```
const employees = [
  { id: 1, name: "Alice", salary: 5000 },
  { id: 2, name: "Bob", salary: 8000 },
  { id: 3, name: "Charlie", salary: 7000 }
];

// Add tax property to each employee
const updatedEmployees = employees.map(employee => ({
  ...employee, // Spread operator to copy existing properties
  tax: employee.salary * 0.1
}));

console.log(updatedEmployees);
/*
Output:
[
  { id: 1, name: "Alice", salary: 5000, tax: 500 },
  { id: 2, name: "Bob", salary: 8000, tax: 800 },
  { id: 3, name: "Charlie", salary: 7000, tax: 700 }
]
*/
```

Explanation:

- Transformation: Each employee object is transformed by adding a tax property.

- Spread Operator: Ensures the original properties are retained while adding a new one.
- Result: Returns a new array with updated objects.

2. reduce

Purpose

reduce combines array elements into a single value, such as a sum, total, or aggregated object.

Syntax

```
array.reduce(callback(accumulator, currentValue, index, array), initialValue)
```

Example: Calculate the Total Cost of Orders

You have a list of orders, and you need to calculate the total cost.

```
const orders = [
  { orderId: 101, product: "Laptop", cost: 1200 },
  { orderId: 102, product: "Mouse", cost: 20 },
  { orderId: 103, product: "Keyboard", cost: 50 }
];

// Calculate total cost
const totalCost = orders.reduce((acc, order) => acc + order.cost, 0);

console.log(totalCost); // Output: 1270
```

Explanation

- Initial Value: acc starts at 0.
- Accumulation: For each order, the cost is added to the acc.
- Result: Returns the final accumulated value (total cost).

3. filter

Purpose

filter creates a new array containing only the elements that satisfy a given condition.

```
array.filter(callback(currentValue, index, array))
```

Example: Filter Products Based on Availability

You have a list of products and want to filter out the ones that are out of stock.

```
const products = [
  { id: 1, name: "Phone", inStock: true },
  { id: 2, name: "Tablet", inStock: false },
  { id: 3, name: "Laptop", inStock: true }
];

// Filter in-stock products
const availableProducts = products.filter(product => product.inStock);

console.log(availableProducts);
/*
Output:
[
  { id: 1, name: "Phone", inStock: true },
  { id: 3, name: "Laptop", inStock: true }
]
*/
```

Explanation

- Condition: product.inStock checks if the product is available.
- Result: Only products with inStock: true are included in the new array.

4. find

Purpose

find returns the first element in the array that satisfies the given condition. If no match is found, it returns undefined.

Syntax

```
array.find(callback(currentValue, index, array))
```

Example: Find a Student by ID

You have a list of students, and you need to find the student with a specific ID.

```
const students = [
  { id: 1, name: "John", age: 18 },
  { id: 2, name: "Sarah", age: 20 },
  { id: 3, name: "Tom", age: 19 }
];
```

```
// Find student with id 2
const student = students.find(student => student.id === 2);

console.log(student); // Output: { id: 2, name: "Sarah", age: 20 }
```

Explanation

- Condition: student.id === 2 identifies the matching student.
- Result: Returns the first matching student object.

5. findIndex

Purpose

findIndex returns the index of the first element in the array that satisfies the condition. If no match is found, it returns -1.

Syntax

```
array.findIndex(callback(currentValue, index, array))
```

Example: Find the Index of a Task by Title

You have a list of tasks, and you want to find the index of the task with the title "Meeting".

```
const tasks = [
  { id: 1, title: "Email", completed: true },
  { id: 2, title: "Meeting", completed: false },
  { id: 3, title: "Call", completed: true }
];

// Find the index of the "Meeting" task
const meetingIndex = tasks.findIndex(task => task.title === "Meeting");

console.log(meetingIndex); // Output: 1
```

Explanation

- Condition: task.title === "Meeting" identifies the task.
- Result: Returns the index of the first matching task (1).

6. Combining filter and map for Complex Queries

Problem:

Extract the names of employees earning more than 50,000, sorted alphabetically.

Code:

```
let employees = [
  { name: "Alice", salary: 70000 },
  { name: "Bob", salary: 45000 },
  { name: "Charlie", salary: 55000 },
  { name: "David", salary: 60000 }
];

let highEarners = employees
  .filter(employee => employee.salary > 50000)
  .map(employee => employee.name)
  .sort();

console.log(highEarners);
// Output: ['Alice', 'Charlie', 'David']
```

Explanation:

- Use filter to keep only employees earning more than 50,000.
- Use map to extract their names.
- Use sort to arrange the names alphabetically.

7. Using find and findIndex for Specific Lookups

Problem:

Find the details and position of the first employee in the "IT" department.

Code:

```
let employees = [
  { name: "Alice", department: "HR" },
  { name: "Bob", department: "IT" },
  { name: "Charlie", department: "Finance" },
  { name: "David", department: "IT" }
];

let firstITEmployee = employees.find(employee => employee.department ===
"IT");
let firstITIndex = employees.findIndex(employee => employee.department
```

```
=== "IT");

console.log(firstITEmployee);  // Output: { name: 'Bob', department: 'IT'
}
console.log(firstITIndex);     // Output: 1
```

Explanation:

- Use find to locate the first employee in the IT department.
- Use findIndex to determine the position of the first IT employee.

References:

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex