1. `filter()`

✅ What it does:

Returns a **new array** containing only the **items that match a condition**.

📦 Syntax:

javascript Copy Edit

```javascript
array.filter(item => condition);
```

🧩 Example:

javascript Copy Edit

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

✅ Use Case in React:

jsx Copy Edit

```jsx
const tasks = [ { id: 1, text: "Buy milk", done: true }, { id: 2, text: "Clean room", done: false
} ]; function TaskList() { return ( <ul> {tasks .filter(task => !task.done) .map(task => ( <li
key={task.id}>{task.text}</li> ))} </ul> ); }
```

🧮 2. `reduce()`

✅ What it does:

Takes an array and **reduces it to a single value** (like a total, sum, object, or even another array).

📦 Syntax:

javascript Copy Edit

```javascript
array.reduce((accumulator, current) => { return newAccumulator; }, initialValue);
```

🧩 Example (Sum):

javascript Copy Edit

```javascript
const numbers = [10, 20, 30]; const total = numbers.reduce((sum, n) => sum + n, 0);
console.log(total); // 60
```

🧩 Example (Count completed tasks):

javascript Copy Edit

```javascript
const tasks = [ { id: 1, done: true }, { id: 2, done: false }, { id: 3, done: true } ]; const
completed = tasks.reduce((count, task) => task.done ? count + 1 : count, 0);
console.log(completed); // 2
```

🧠 Summary Table

Method	Purpose	Returns	Example Use
`filter()`	Filters items by condition	New array	Show only completed tasks
`reduce()`	Combines items into one value	Single value	Total price, count items, sum nums