

Filter Method: Works on ALL Types of Arrays

The `filter()` method works on **any array**, not just array of objects. Let me show you examples:

1. Filter on Simple 1D Array (Numbers)

javascript

```
JavaScript
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

// Filter even numbers
const evenNumbers = numbers.filter(num => num % 2 === 0);
console.log(evenNumbers); // [2, 4, 6, 8, 10]

// Filter numbers greater than 5
const bigNumbers = numbers.filter(num => num > 5);
console.log(bigNumbers); // [6, 7, 8, 9, 10]
```

2. Filter on Simple 1D Array (Strings)

javascript

```
JavaScript
const fruits = ["apple", "banana", "cherry", "apricot", "blueberry"];

// Filter fruits that start with 'a'
const aFruits = fruits.filter(fruit => fruit.startsWith('a'));
console.log(aFruits); // ["apple", "apricot"]

// Filter fruits with more than 6 letters
const longFruits = fruits.filter(fruit => fruit.length > 6);
console.log(longFruits); // ["blueberry"]
```

3. Filter on Array of Objects

javascript

JavaScript

```
const products = [
  {name: "PHANTOM X", price: 349, category: "headphones"},
  {name: "WAVE PRO", price: 599, category: "speakers"},
  {name: "PULSE MINI", price: 179, category: "earbuds"}
];

// Filter products cheaper than $200
const affordable = products.filter(product => product.price < 200);
console.log(affordable);
// [{name: "PULSE MINI", price: 179, category: "earbuds"}]

// Filter only headphones
const headphones = products.filter(product => product.category === "headphones");
console.log(headphones);
// [{name: "PHANTOM X", price: 349, category: "headphones"}]
```

How Filter Works

javascript

JavaScript

```
array.filter(callback function)
```

For each element in the array:

1. Pass the element to the callback function
2. If callback returns **true** → keep the element
3. If callback returns **false** → remove the element
4. Return a new array with only the kept elements

Visual Example

javascript

JavaScript

```
const ages = [12, 18, 25, 30, 15, 45];

const adults = ages.filter(age => age >= 18);

// What happens:
12 >= 18? false → ❌ remove
18 >= 18? true → ✅ keep
25 >= 18? true → ✅ keep
30 >= 18? true → ✅ keep
15 >= 18? false → ❌ remove
45 >= 18? true → ✅ keep

// Result: [18, 25, 30, 45]
```

Comparison Table

Array Type	Example	Filter Usage
Numbers	[1, 2, 3, 4, 5]	<code>arr.filter(num => num > 3)</code>
Strings	["a", "b", "c"]	<code>arr.filter(str => str !== "b")</code>
Booleans	[true, false, true]	<code>arr.filter(bool => bool === true)</code>
Objects	[{id: 1}, {id: 2}]	<code>arr.filter(obj => obj.id === 1)</code>

Interactive Demo

Let me create a file showing all these examples with `console.log`:

Created a file, read a file

Summary

The **filter()** method works on ALL types of arrays:

✓ **1D Arrays (Simple Arrays):**

- Numbers: `[1, 2, 3, 4, 5]`
- Strings: `["apple", "banana", "cherry"]`
- Booleans: `[true, false, true]`

✓ **Array of Objects:**

- `[{id: 1, name: "Product"}, {id: 2, name: "Item"}]`

✓ **Mixed Arrays:**

- `[1, "hello", true, {id: 1}]`

Key Points:

1. **Filter works on ANY array** - it doesn't care what type of data is inside
2. **Returns a new array** - original array stays unchanged
3. **Callback function decides** - if it returns `true`, keep the element; if `false`, remove it