The filter() method in JavaScript is used to create a new array containing only the elements of the original array that satisfy a specified condition (i.e., for which the callback function returns true). It does not modify the original array.

---

### \*Syntax\*

javascript

array.filter(callback(element, index, array), thisArg);

- \*callback\*: A function that is called for each element. It receives the following arguments:

- \*element\*: The current element being processed.

- \*index\* (Optional): The index of the current element.

- \*array\* (Optional): The array filter() was called on.

- \*thisArg\* (Optional): Value to use as this when executing the callback.

---

### \*Examples of filter()\*

---

### \*1. Filtering Numbers\*

Filter out numbers greater than a certain value.

javascript

let numbers = [1, 2, 3, 4, 5];

let filtered = numbers.filter(num => num > 2);

console.log(filtered); // Output: [3, 4, 5]

---

### \*2. Filtering Even Numbers\*

Use a condition to filter even numbers.

javascript

let numbers = [1, 2, 3, 4, 5];

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

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

---

### \*3. Filtering Strings\*

Filter strings based on their length.

javascript

let words = ["apple", "banana", "pear", "kiwi"];

let longWords = words.filter(word => word.length > 4);

console.log(longWords); // Output: ["apple", "banana"]

---

### \*4. Filtering Objects\*

Extract objects that meet specific criteria.

javascript

let products = [

{ name: "Laptop", price: 1000 },

{ name: "Phone", price: 500 },

{ name: "Tablet", price: 300 }

];

let expensiveProducts = products.filter(product => product.price > 400);

console.log(expensiveProducts);

// Output: [{ name: "Laptop", price: 1000 }, { name: "Phone", price: 500 }]

---

### \*5. Using Index in filter()\*

Filter elements based on their index.

javascript

let numbers = [10, 20, 30, 40, 50];

let filtered = numbers.filter((num, index) => index % 2 === 0);

console.log(filtered); // Output: [10, 30, 50]

---

### \*6. Filtering Boolean Values\*

Remove false, null, 0, undefined, and NaN values.

javascript

let mixed = [0, 1, false, 2, "", 3, null, "hello"];

let truthy = mixed.filter(Boolean);

console.log(truthy); // Output: [1, 2, 3, "hello"]

---

### \*7. Filtering Nested Arrays\*

Filter nested arrays based on conditions.

javascript

let matrix = [[1, 2], [3, 4, 5], [6]];

let filtered = matrix.filter(arr => arr.length > 2);

console.log(filtered); // Output: [[3, 4, 5]]

---

### \*8. Filtering Unique Values\*

Use filter() to find unique values in an array.

javascript

let numbers = [1, 2, 2, 3, 4, 4, 5];

let unique = numbers.filter((num, index, arr) => arr.indexOf(num) === index);

console.log(unique); // Output: [1, 2, 3, 4, 5]

---

### \*9. Chaining with Other Methods\*

Combine filter() with other array methods like map().

javascript

let numbers = [1, 2, 3, 4, 5];

let result = numbers.filter(num => num % 2 === 0).map(num => num \* 2);

console.log(result); // Output: [4, 8]

---

### \*10. Filtering Based on Custom Conditions\*

Filter elements using a custom function.

javascript

let students = [

{ name: "Alice", grade: 85 },

{ name: "Bob", grade: 70 },

{ name: "Charlie", grade: 90 }

];

let passedStudents = students.filter(student => student.grade >= 80);

console.log(passedStudents);

// Output: [{ name: "Alice", grade: 85 }, { name: "Charlie", grade: 90 }]

---

### Summary

- The filter() method is useful for creating a subset of elements from an array based on specific conditions.

- It \*does not modify the original array\* but returns a new one.

- The callback function must return true to keep an element in the new array.