The sort() method in JavaScript is used to sort the elements of an array in place and returns the sorted array. By default, it sorts the elements as strings in ascending order. However, you can provide a custom sorting function to sort the elements based on different criteria (e.g., numbers, objects).

---

### \*Syntax\*

javascript

array.sort(compareFunction);

- \*compareFunction\* (optional): A function that defines the sort order. It takes two arguments and should return:

- A negative value if the first argument is less than the second.

- A positive value if the first argument is greater than the second.

- 0 if they are equal.

---

### \*Examples of sort()\*

---

### \*1. Sorting Strings Alphabetically\*

By default, sort() sorts elements as strings.

javascript

let fruits = ["banana", "apple", "cherry", "date"];

fruits.sort();

console.log(fruits); // Output: ["apple", "banana", "cherry", "date"]

---

### \*2. Sorting Numbers (Incorrect Order)\*

By default, sort() converts numbers to strings, which can lead to unexpected results.

javascript

let numbers = [10, 2, 33, 4];

numbers.sort();

console.log(numbers); // Output: [10, 2, 33, 4] (Incorrect sorting)

---

### \*3. Sorting Numbers Correctly\*

To sort numbers in ascending or descending order, you need to pass a custom comparison function.

javascript

let numbers = [10, 2, 33, 4];

// Ascending order

numbers.sort((a, b) => a - b);

console.log(numbers); // Output: [2, 4, 10, 33]

// Descending order

numbers.sort((a, b) => b - a);

console.log(numbers); // Output: [33, 10, 4, 2]

---

### \*4. Sorting Strings in Reverse Order\*

You can reverse the order of strings with a custom comparison function.

javascript

let fruits = ["banana", "apple", "cherry", "date"];

fruits.sort((a, b) => b.localeCompare(a));

console.log(fruits); // Output: ["date", "cherry", "banana", "apple"]

---

### \*5. Sorting Arrays of Objects by Property\*

You can sort arrays of objects by a specific property, such as a name or price.

javascript

let products = [

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

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

{ name: "Tablet", price: 300 }

];

// Sort by price in ascending order

products.sort((a, b) => a.price - b.price);

console.log(products);

// Output: [{ name: "Tablet", price: 300 }, { name: "Phone", price: 500 }, { name: "Laptop", price: 1000 }]

// Sort by name in alphabetical order

products.sort((a, b) => a.name.localeCompare(b.name));

console.log(products);

// Output: [{ name: "Laptop", price: 1000 }, { name: "Phone", price: 500 }, { name: "Tablet", price: 300 }]

---

### \*6. Sorting Case-Insensitive Strings\*

To perform a case-insensitive sort of strings, use localeCompare():

javascript

let words = ["banana", "Apple", "cherry", "apple"];

words.sort((a, b) => a.toLowerCase().localeCompare(b.toLowerCase()));

console.log(words); // Output: ["apple", "Apple", "banana", "cherry"]

---

### \*7. Sorting Dates\*

You can sort an array of Date objects:

javascript

let dates = [

new Date(2023, 11, 24),

new Date(2021, 0, 1),

new Date(2022, 6, 15)

];

// Sort dates in ascending order

dates.sort((a, b) => a - b);

console.log(dates);

// Output: [2021-01-01T00:00:00.000Z, 2022-06-15T00:00:00.000Z, 2023-12-24T00:00:00.000Z]

---

### \*8. Sorting Arrays of Strings by Length\*

You can sort strings by their length:

javascript

let words = ["banana", "apple", "cherry", "date"];

words.sort((a, b) => a.length - b.length);

console.log(words); // Output: ["date", "apple", "banana", "cherry"]

---

### \*9. Sorting in Descending Order Using reverse()\*

After sorting in ascending order, you can reverse the array:

javascript

let numbers = [10, 2, 33, 4];

numbers.sort((a, b) => a - b).reverse();

console.log(numbers); // Output: [33, 10, 4, 2]

---

### \*10. Sorting Without Modifying the Original Array\*

If you need to avoid modifying the original array, make a copy of the array before using sort():

javascript

let numbers = [10, 2, 33, 4];

let sortedNumbers = [...numbers].sort((a, b) => a - b);

console.log(sortedNumbers); // Output: [2, 4, 10, 33]

console.log(numbers); // Original array remains the same: [10, 2, 33, 4]

---

### Summary

- The sort() method sorts the array in place and returns the sorted array.

- By default, sort() converts elements to strings and sorts them lexicographically.

- To sort numbers or perform custom sorting, you need to use a comparison function.