# JS ARRAY SORT

## Sorting an Array

**The sort() method sorts an array alphabetically:**

```
<p>The sort() method sorts an array alphabetically:</p>

<p id="demo1"></p>
<p id="demo2"></p>

<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;

fruits.sort();
document.getElementById("demo2").innerHTML = fruits;
</script>
```

**Output**

```
Banana,Orange,Apple,Mango
Apple,Banana,Mango,Orange
```

**Reversing an Array**

The reverse() method reverses the elements in an array.

You can use it to sort an array in descending order:

```
<p>The reverse() method reverses the elements in an array.</p>
<p>By combining sort() and reverse() you can sort an array in descending order:</p>

<p id="demo1"></p>
<p id="demo2"></p>

<script>
// Create and display an array:
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;

// First sort the array
fruits.sort();

// Then reverse it:
fruits.reverse();

document.getElementById("demo2").innerHTML = fruits;
</script>
```

**Output**

```
Banana,Orange,Apple,Mango
Orange,Mango,Banana,Apple
```

**Numeric Sort**

By default, the sort() function sorts values as strings.
This works well for strings ("Apple" comes before "Banana").
However, if numbers are sorted as strings, "25" is bigger than "100", because "2" is bigger than "1".
Because of this, the sort() method will produce incorrect result when sorting numbers.

**You can fix this by providing a compare function:**

```
<p>Sort the array in ascending order:</p>

<p id="demo1"></p>
<p id="demo2"></p>

<script>
const points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo1").innerHTML = points;

points.sort(function(a, b){return a - b});
document.getElementById("demo2").innerHTML = points;
</script>
```

**Output**

**Sort the array in ascending order:**

```
40,100,1,5,25,10
1,5,10,25,40,100
```

**Use the same trick to sort an array descending:**

```
<p>Sort the array in descending order:</p>

<p id="demo1"></p>
<p id="demo2"></p>

<script>
const points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo1").innerHTML = points;

points.sort(function(a, b){return b - a});
document.getElementById("demo2").innerHTML = points;
</script>
```

**Output**

**Sort the array in descending order:**

```
40,100,1,5,25,10
100,40,25,10,5,1
```

**The Compare Function**

- The purpose of the compare function is to define an alternative sort order.
- The compare function should return a negative, zero, or positive value, depending on the arguments:
- function(a, b){return a - b}
- When the sort() function compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value.
- If the result is negative, a is sorted before b.
- If the result is positive, b is sorted before a.
- If the result is 0, no changes are done with the sort order of the two values.

**Example:**

The compare function compares all the values in the array, two values at a time (a, b).
When comparing 40 and 100, the sort() method calls the compare function(40, 100).
The function calculates 40 - 100 (a - b), and since the result is negative (-60), the sort function will sort 40 as a value lower than 100.

**You can use this code snippet to experiment with numerically and alphabetically sorting:**

```
<p>Click the buttons to sort the array alphabetically or numerically.</p>

<button onclick="myFunction1()">Sort Alphabetically</button>
<button onclick="myFunction2()">Sort Numerically</button>

<p id="demo"></p>

<script>
const points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = points;

function myFunction1() {
  points.sort();
  document.getElementById("demo").innerHTML = points;
}
function myFunction2() {
  points.sort(function(a, b){return a - b});
  document.getElementById("demo").innerHTML = points;
}
</script>
```

**Output:**

**Sort alphabetically**
```
1,10,100,25,40,5
```
**Sort numerically**
```
1,5,10,25,40,100
```

**Sorting an Array in Random Order**

```
<p>Click the button (again and again) to sort the array in random order.</p>

<button onclick="myFunction()">Try it</button>
<p id="demo"></p>

<script>
const points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = points;

function myFunction() {
  points.sort(function(){return 0.5 - Math.random()});
  document.getElementById("demo").innerHTML = points;
}
</script>
```

**Output:**

```
10,100,40,1,5,25
```

**The Fisher Yates Method**

The above example, array.sort(), is not accurate. It will favor some numbers over the others. The most popular correct method, is called the Fisher Yates shuffle, and was introduced in data science as early as 1938!

**In JavaScript the method can be translated to this:**

```
<p>Click the button (again and again) to sort the array in random order.</p>

<button onclick="myFunction()">Try it</button>
<p id="demo"></p>

<script>
const points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = points;

function myFunction() {
  for (let i = points.length -1; i > 0; i--) {
    let j = Math.floor(Math.random() * (i+1));
    let k = points[i];
    points[i] = points[j];
    points[j] = k;
  }
  document.getElementById("demo").innerHTML = points;
}
</script>
```

**output**

```
25,40,100,10,5,1
```

**Find the Highest (or Lowest) Array Value**

There are no built-in functions for finding the max or min value in an array.
However, after you have sorted an array, you can use the index to obtain the highest and lowest values.

**Sorting ascending:**

```
<p>The lowest number is <span id="demo"></span>.</p>

<script>
const points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a-b});
document.getElementById("demo").innerHTML = points[0];
</script>
```

**Output:**

```
The lowest number is 1.
```

```
<p>The highest number is <span id="demo"></span>.</p>
```

Sorting descending:

```
<script>
const points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return b-a});
document.getElementById("demo").innerHTML = points[0];
</script>
```

**Output:**

```
The highest number is 100.
```

**Using Math.max() on an Array**

Math.max.apply(null, [1, 2, 3]) is equivalent to Math.max(1, 2, 3).

You can use Math.max.apply to find the highest number in an array:

```
<p>The highest number is <span id="demo"></span>.</p>

<script>
const points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = myArrayMax(points);

function myArrayMax(arr) {
  return Math.max.apply(null, arr);
}
</script>
```

**Output**

```
The highest number is 100.
```

**Using Math.min() on an Array**

You can use Math.min.apply to find the lowest number in an array:

```
<p>The lowest number is <span id="demo"></span>.</p>

<script>
const points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = myArrayMin(points);

function myArrayMin(arr) {
  return Math.min.apply(null, arr);
}
</script>
```

**Output**

```
The lowest number is 1.
```

**My Min / Max JavaScript Methods**

The fastest solution is to use a "home made" method.
This function loops through an array comparing each value with the highest value found:

```
<p>The highest number is <span id="demo"></span>.</p>

<script>
const points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = myArrayMax(points);

function myArrayMax(arr) {
  let len = arr.length;
  let max = -Infinity;
  while (len--) {
    if (arr[len] > max) {
      max = arr[len];
    }
  }
  return max;
}
</script>
```

**Output**

```
The highest number is 100.
```

This function loops through an array comparing each value with the lowest value found:

```
<p>The lowest number is <span id="demo"></span>.</p>

<script>
const points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = myArrayMin(points);

function myArrayMin(arr) {
  let len = arr.length;
  let min = Infinity;
  while (len--) {
    if (arr[len] < min) {
      min = arr[len];
    }
  }
  return min;
}
</script>
```

**Output**

```
The lowest number is 1.
```

**Sorting Object Arrays**

JavaScript arrays often contain objects:

```
const cars = [
  {type:"Volvo", year:2016},
  {type:"Saab", year:2001},
  {type:"BMW", year:2010}
];
```

Even if objects have properties of different data types, the sort() method can be used to sort the array.
The solution is to write a compare function to compare the property values

```
<p>Sort car objects on age:</p>

<p id="demo"></p>

<script>
const cars = [
  {type:"Volvo", year:2016},
  {type:"Saab", year:2001},
  {type:"BMW", year:2010}
];

displayCars();

cars.sort(function(a, b){return a.year - b.year});
displayCars();

function displayCars() {
  document.getElementById("demo").innerHTML =
  cars[0].type + " " + cars[0].year + "<br>" +
  cars[1].type + " " + cars[1].year + "<br>" +
  cars[2].type + " " + cars[2].year;
}
</script>
```

**Output**

```
Sort car objects on age:
Saab 2001
BMW 2010
Volvo 2016
```

**Comparing string properties is a little more complex:**

```
<p>Click the buttons to sort car objects on type.</p>

<button onclick="myFunction()">Sort</button>
<p id="demo"></p>

<script>
const cars = [
  {type:"Volvo", year:2016},
  {type:"Saab", year:2001},
  {type:"BMW", year:2010}
];

displayCars();

function myFunction() {
  cars.sort(function(a, b){
    let x = a.type.toLowerCase();
    let y = b.type.toLowerCase();
    if (x < y) {return -1;}
    if (x > y) {return 1;}
    return 0;
  });
  displayCars();
}

function displayCars() {
  document.getElementById("demo").innerHTML =
  cars[0].type + " " + cars[0].year + "<br>" +
  cars[1].type + " " + cars[1].year + "<br>" +
  cars[2].type + " " + cars[2].year;
}
</script>
```

**Output:**

```
BMW 2010
Saab 2001
Volvo 2016
```