JavaScript Sorting Arrays

Sorting an Array

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

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();        // Sorts the elements of fruits
```

Reversing an Array

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

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

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();        // First sort the elements of fruits
fruits.reverse();     // Then reverse the order of the elements
```

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**:

Example

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a - b});
```

Use the same trick to sort an array descending:

Example

```
var points = [40, 100, 1, 5, 25, 10];
```

```
points.sort(function(a, b){return b - a});
```

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:

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

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

<script>
var 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>
```

Sorting an Array in Random Order

Example

```
var points = [40, 100, 1, 5, 25, 10];
```

```
points.sort(function(a, b){return 0.5 - Math.random()});
```

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:

Example

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a - b});
// now points[0] contains the lowest value
// and points[points.length-1] contains the highest value
```

Sorting descending:

Example

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return b - a});
// now points[0] contains the highest value
// and points[points.length-1] contains the lowest value
```

Sorting a whole array is a very inefficient method if you only want to find the highest (or lowest) value.

Using Math.max() on an Array

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

Example

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

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

Using Math.min() on an Array

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

Example

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

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

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:

Example (Find Max)

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

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

Example (Find Min)

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

```
}
```

Sorting Object Arrays

JavaScript arrays often contain objects:

Example

```
var 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:

Example

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

Comparing string properties is a little more complex:

Example

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

Exercise:

Use the correct Array method to sort the `fruits` array alphabetically.

```
var fruits = ["Banana", "Orange", "Apple", "Kiwi"];
             ;
```

Submit Answer »

Start the Exercise