Sorting

Sorting is the process of rearranging elements in a particular order – usually ascending or descending. We will use the .sort( ) method to sort array values.

```
pets = ['dog', 'bird', 'fish', 'cat'];

let petSort = pets.sort();

petSort would hold ['bird', 'cat', 'dog', 'fish']
```

If we have a simple string array, the .sort( ) method alphabetizes our values. But with numbers and object properties, you need a compare function.

That's because JavaScript's default .sort() converts values to strings -- which means 10 would be seen as smaller than 2.

```
const numbers = [10, 2, 30];
console.log(numbers.sort());
// → [10, 2, 30]
```

This is a compare function that can be reused for many sorting scenarios:

```
function compareFn(a,b) {
    if (a < b) {
      return –1;
    } else if (a > b) {
      return 1;
    }
   return 0;
  }

const sortList = list.sort(compareFn)
```

This is still using the .sort( ) method, but there is an argument that calls a function. In this example that is compareFn. The parameters a and b represent the first two value to compare. The function will be called repeatedly, comparing each value to the next until all are sorted.

The ‘return -1’ and ‘return 1’ determine whether values are moved up or down in the sort.

You’ll also want to make sure your strings are all converted to lowercase, since uppercase and lowercase letters are sorted differently in JavaScript (and most programming languages). This is due to character encoding — specifically Unicode or ASCII.

For example, in this array, some pets start with uppercase letters. In Unicode, uppercase letters have lower character codes and come before lowercase letters:

```
petList = ['Fish', 'ferret', 'dog', 'Cat']
```

Using .sort( ) directly would give**:** **['Cat', 'Fish', 'dog', 'ferret']**

To sort without case issues, convert everything to lowercase using .toLowerCase(). Here, we use .map() to make a new array of all-lowercase pets:

```
let lowerSort = petList.map(pet => pet.toLowerCase()).sort();
```

Then you'd get: [**'cat', 'dog', 'ferret', 'fish']**

Sorting Objects

Let’s look at sorting an array of objects. Here we have an array of hikes, each with a name and distance:

```
const hikes = [
  { name: "Bechler Falls", distance: "3 miles" },
  { name: "Teton Canyon", distance: "5 miles" },
  { name: "Denanda Falls", distance: "7 miles" },
  { name: "Menan Butte", distance: "3.4 miles" }
];
```

We can sort this array in multiple ways — by name or by distance. To do that, we use a **custom compare function** that targets a specific property.

For example, to sort by distance:

```
function compareHikes(a,b) {
    if (a.distance < b.distance) {
      return -1;
    } else if (a.distance > b.distance) {
      return 1;
    }
   return 0;
  }

   let sortedHikes = hikes.sort(compareHikes);

   console.log(sortedHikes);
```

This will display the hikes from shortest to longest distance.

So, there we have the .sort( ) method being used in different ways with different types of arrays.