Arrays Methods

length

length returns how many elements are in the array.

This is a property, NOT a function (you can tell because we type **length**, not **length()**.

```
const arr14 = [1, 2, 3, 4];
console.log(arr14.length); // 4
console.log(arr14[arr14.length]); // undefined
console.log(arr14[arr14.length - 1]);
```

slice

slice makes a copy of an array.

We can use it to copy the entire array

```
const arr15 = [1, 2, 3, 4];
const copy = arr15.slice();
console.log(arr15); // [1,2,3,4];
```

slice

Alternatively, you can pass in two arguments to slice. Like splice, the first argument indicates the starting index of the subarray you want.

```
const arr16 = [7, 6, 5, 4, 3, 2];
const copya = arr16.slice(1, 2);
console.log(copya); // [6]
const copyb = arr16.slice(2, 5);
console.log(copyb); // [5, 4, 3]
const copyc = arr16.slice(2, 1);
console.log(copyc); // []
```

The second argument indicates the ending index. The subarray you get will consist of all the values starting from the starting index and going up to (but not including) the ending index

concat

concat joins two arrays together.

```
const arr18 = [1, 2, 3];
const arr19 = [4, 5, 6];
const combined1 = arr18.concat(arr19);
console.log(combined1); // [1,2,3,4,5,6]
```

concat

You can pass multiple arrays into concat
and it will still return a single array

```
const arr20 = ['a', 'b', 'c'];
const arr21 = ['d', 'e', 'f'];
const arr22 = ['g', 'h', 'i'];
const combined2 = arr20.concat(arr21, arr22);
console.log(combined2);
// ['a','b','c','d','e','f','g','h','i'];
```

concat

Any comma-separated list of values can be concatenated with the original array:

```
const openingWords = ['It', 'was', 'a'];
const moreOpeningWords = openingWords.concat('dark', 'and', 'stormy', 'night');
console.log(moreOpeningWords);
// ['It', 'was', 'a', 'dark', 'and', 'stormy', 'night']
```

join

join joins elements of an array into a string separated by whatever you pass in as an argument to join.

This argument is frequently referred to as a delimiter.

```
const arr23 = ['Hello', 'World'];
const combined3 = arr23.join(' ‘);
console.log(combined3); // 'Hello World'
```

```
var arr24 = ['I', 'have', 'a', 'big', 'announcement'];
const combined4 = arr24.join('! ') + '!';
console.log(combined4);
// 'I! have! a! big! announcement!'
```

indexOf

indexOf finds the first index of the element passed in (starting from the left).

If the element is not found, it returns -1

```
const arr25 = [1, 2, 3, 4, 5, 4, 4];
console.log(arr25.indexOf(2)); // 1
console.log(arr25.indexOf(3)); // 2
console.log(arr25.indexOf(1));
                // 0 - remember, arrays are zero indexed
console.log(arr25.indexOf(4));
                // 3 - indexOf stops once it finds the first 4.
console.log(arr25.indexOf(10)); // -1
```

indexOf

this function commonly used to check if an element is in an array or not

```
const moviesIKnow = [
  'Waynes World',
  'The Matrix',
  'Anchorman',
  'Bridesmaids',
];

var yourFavoriteMovie = prompt('Whats your favorite movie?');
if (moviesIKnow.indexOf(yourFavoriteMovie) > -1) {
  alert('Oh, cool, Ive heard of ' + yourFavoriteMovie + '!');
} else {
  alert('I havent heard of ' + yourFavoriteMovie + '. Ill check it out.');
}
```

lastIndexOf

lastIndexOf works just like indexOf, but starts searching from the end of the array rather than the beginning

```
const arr26 = [1, 2, 3, 4, 5, 4, 4];
console.log(arr26.indexOf(4)); // 3
console.log(arr26.lastIndexOf(4));
              // 6 - this one is different now as it starts from the end!
console.log(arr26.lastIndexOf(10));
              // -1 - still returns -1 if the value is not found in the array
```