Most common array operations

Arrays are commonly used data structures in JavaScript so we have collected the most useful operations on arrays.

Create an array

Easiest way to do is simply put **[]** in your code:

```
const array = [];
```

This is going to create an empty array, but it's possible to initialize an array with any number of elements (and any type).
Arrays in JavaScript are non heterogeneous (so one array can contains multiple types of values).

```
const numArray = [1, 2, 3];
const mixedArray = [{name: 'Lisa', age: 28}, false, 'dog'];
```

Length of an array

The **length** property on an array will return it:

```
const array = [3, 4, 5];

console.log(array.length); // 3
```

Add new items

It's possible add new elements in multiple ways, but the most common is the push method, which can insert one (or multiple) element(s) at a time at the end of an array.

```
const array = [3, 4, 5];

array.push(7);
array.push(8, 9, 20);
```

If we want to add elements at the beginning of an array, we can use the unshift method.

As a newer solution, we can use the spread operator on arrays:

```
const array = [3, 4, 5];

array = [...array, 7]; // Insert an element at the end
array = [8, 9, 20, ...array]; // Insert an element at the beginning
```

Note that this method creates a new array, while **push** and **unshift** were operate on the reference.

Access an element at an index

We can access an element with the index of it. Note that indexing in JavaScript starts at 0.

```
const array = [3, 4, 5];

console.log(array[0]); // 3
console.log(array[array.length - 1]); // last item, 5
console.log(array[10]); // there's no 11 items in the array, so it
returns undefined
```

A value can be replaced at an index:

```
array[0] = 7;
```

Delete an element

There's multiple ways to do it, but the most common is the splice method.

```
const array = [3, 4, 5];

array.splice(1, 1); // Első paraméter: a törlendő elem(ek) index-e,
második paraméter: hány elemet akarunk törölni

console.log(array);
```

If we want to remove the first or the last element of an array, then we can use the shift and pop methods.

Iterate through an array

It's possible to use while or for loop for this purpose, but the newer, more functional approach is to use the forEach method, which expects a function as a parameter and it will calls this function for every item in the array.

```
const array = ['apple', 'orange', 'melon', 'banana'];

array.forEach((element, index) => {
 console.log(`${index + 1}. ${element}`);
});
```

Further array operations

List of array operations: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array