**Callback functions**

A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete a routine or action. Here is a quick example:

function greeting(name) {

alert('Hello ' + name);

}

function processUserInput(callback) {

var name = prompt('Please enter your name.');

callback(name);

}

processUserInput(greeting);

All the methods below take a callback function as one of their arguments. Often, the callback function specifies the parameters your want to use in the outer function.

**array.forEach()**

The forEach() method executes a given function once for each array element.

const array1 = ['a', 'b', 'c'];

array1.forEach(element => console.log(element));

// expected output: "a"

// expected output: "b"

// expected output: "c"

**array.filter()**

The filter() method creates a new array with all the elements that pass the test implemented by the given function.

const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];

const result = words.filter(word => word.length > 6);

console.log(result);

// expected output: Array ["exuberant", "destruction", "present"]

**array.sort()**

The sort() method sorts the elements of an array in place and returns the sorted array. The default sort of the order is ascending.

const months = ['March', 'Jan', 'Feb', 'Dec'];

months.sort();

console.log(months);

// expected output: Array ["Dec", "Feb", "Jan", "March"]

const array1 = [1, 30, 4, 21, 100000];

array1.sort();

console.log(array1);

// expected output: Array [1, 100000, 21, 30, 4]

**array.map()**

The map() method creates a new array, populated with the results of calling a given function on every element in the original array.

const array1 = [1, 4, 9, 16];

//Pass a function to map, in this case, a function that multiplies each element in the original array by 2.

const map1 = array1.map(x => x \* 2);

console.log(map1);

//Now a new array is returned and the expected output of each element is multiplied by 2: Array [2, 8, 18, 32]

**array.reduce()**

The reduce() method executes a reducer function (that you write) on each element of the array, resulting in single output value. In this case, the reducer function adds the current number in the array to the totaled numbers thus far.

const array1 = [1, 2, 3, 4];

const reducer = (accumulator, currentValue) => accumulator + currentValue;

// 1 + 2 + 3 + 4

console.log(array1.reduce(reducer));

// expected output: 10

// 5 + 1 + 2 + 3 + 4

console.log(array1.reduce(reducer, 5));

// expected output: 15

-[Source: MDN](https://developer.mozilla.org/en-US/)

Source:

MITxPro

https://executive-ed.xpro.mit.edu/professional-certificate-coding?utm\_source=Google&utm\_medium=c&utm\_term=%2Bmit%20%2Bcoding&utm\_location=9007574&utm\_campaign=B-365D\_US\_GG\_SE\_PCC\_Brand&utm\_content=MIT-Coding\_\_\_School\_Duration&gclid=Cj0KCQjw3duCBhCAARIsAJeFyPVYZJwvrGBn\_93CMgrokibm87uQ\_OnDXxiBDYwzsRiQcPqa2\_vVawoaAiLCEALw\_wcB