find Method

- find method returns the value of the first element inside an array that passes a certain test in a callback function.

```
- const scores = [10,5,0,40,30,10,90,70];
const firstHighscore = scores.find((score) => {
        return score > 50;
});
console.log(firstHighscore);
```

↓

go

Here we get the first value/element from an array which satisfies condition/test present or given in callback function.

Also this is also valid because we have only one parameter and only one return type so we can shorten this to

```
const scores = [10,5,0,40,30,10,90,70];
const firstHighscore = scores.find(score => score > 50);
console.log(firstHighscore);
```

↓

go

Sort Method

- Sort method is destructive i.e. it changes the original array

- // example-01 - sorting strings

```
const names = ['maria', 'shaun', 'chun-li', 'yoshi', 'luigi'];
console.log(names);
names.sort();
```

(5) ["chun-li", "luigi", "maria", "shaun", "yoshi"]

- // example-02 - sorting numbers

```
const scores = [10, 50, 20, 5, 35, 70, 45];
scores.sort();
console.log(scores);
```

(7) [10, 20, 35, 45, 5, 50, 70]

this is wrong sorted, what happens here is it is sorted only first digit of according to first digit of number.

10, 20, 35, 45, 5, 50, 70

here it is looking at first digit of number only.
i.e. sorted according to first digit of numbers.

- const names = ['maeio', 'shaun', 'chun-li', 'yoshi', 'luigi'];
names.reverse();
console.log(names);

↓

(5) ["luigi", "yoshi", "chun-li", "shaun", "maeio"]

↓

Here what happens is our array is reversed.

Note that here our array is reversed, not in descending order we get!

- const scores = [10, 50, 20, 5, 35, 70, 45];
scores.reverse();
console.log(scores);

↓

(7) [45, 70, 35, 5, 20, 50, 10]

↓

Here our arrays of numbers is reversed.

Date

```
// example-3-sorting objects
const players = [
    {name: 'mario', score: 20},
    {name: 'luigi', score: 10},
    {name: 'chun-li', score: 50},
    {name: 'yoshi', score: 30},
    {name: 'shaun', score: 70}
];
players.sort((a,b) => {
    if(a.score > b.score){
        return -1;
    }else if(b.score > a.score){
        return 1;
    }else{
        return 0;
    }
});
console.log(players);
```

→ (5) [{..}, {..}, {..}, {..}, {..}]

- 0: {name: "shaun", score: 70}
- 1: {name: "chun-li", score: 50}
- 2: {name: "yoshi", score: 30}
- 3: {name: "mario", score: 20}
- 4: {name: "luigi", score: 10}

Here what happens is our array objects are sorted by property score
i.e. 70, 50, 30, 20, 10

- instead of weiting all 9 code lines, we can write just single line and we get result same.

```
players.sort((a,b) => b.score - a.score);
```

↓

this does exactly the same as on previous page. this is the shorter code for us.

- ```
const scores = [10,50,20,5,35,70,45];
scores.sort((a,b) => b-a);
console.log(scores);
```

→ (7) [70,50,45,35,20,10,5].

Here we get all elements/numbers in such a way that at first we get biggest number and at last smallest.

- ```
const scores = [10,50,20,5,35,70,45];
scores.sort((a,b) => a-b);
console.log(scores);
```

→ (7) [5,10,20,35,45,50,70]

Here we get all numbers/elements in such a way that at first we get smallest number and at last biggest number.