3 Arrays

3.1 Keeping track of our games in an array

Up until now we've been keeping track of our games using specific variables for each game. This means that in our function *printAllGames* we must call *addStatus* explicitly for every game. We can automate this by storing a list of our games in an array and then looping over each element.

```
const animals = [
  animal1,
  animal2
];

for (const animal of animals) {
  console.log(animal.name);
}
```

Once we have our games stored in an array, we can loop over the array and execute some functionality on every element.

3.1.1 Exercise: Automatically show all games

Create an array, "games", that contains all your existing games.
Update the printAllGames function to use a for..of loop. Using this for..of loop, call the addStatus function for each element in the "games" array.

Evaluation criteria: You should have in the games.js file:

- ❑ an array, "games", that contains all your games
- ❑ updated the printAllGames function to use a for..of loop
- ❑ only use 1 addStatus call (which happens inside the loop)

3.2 Updating existing functionality

3.2.1 Exercise: Updating the average rating of all your games using arrays

Now that we have an array containing all our games, we can automatically calculate the average rating using a for loop. This way, we don't have to update the getAverageRating function every time we add or remove a game. When calculating the average age of our farm animals, we use a simple for-loop:

```
let result = 0;
for (const animal of animals) {
  result += animal.age;
}
return result / animals.length;
```

Calculate and show the average rating of all the games by looping over the "games" array.

The result should look the same as before, but with an average rating calculated using a loop.

If you get a big number with a lot of decimals, you can set the number of decimals using the built-in JavaScript toFixed function.

Some statistics ...

Average rating: 4.4

Fifa23 is game with the highest rating: 7

Evaluation criteria: You should have updated in the file game.js

- ❑ The function getAverageRating to calculating the average rating using a loop

The average result on the page should be

- ❑ 4.4

You can now add new games to your "games" array to see that your page will automatically update all the elements without you having to write additional code.

3.2.2 Exercise: Showing game with the highest rating using arrays

We've already updated some parts of our code to work with the new "games" array, but we still need to update our getHighestRating function to do this as well.
We can get the highest rating by using a loop as well. Make sure that no error is thrown when the "games" array is empty. Add an if statement that displays a message if that is the case, instead of getting the highest rating.

Update the getHighestRating function to get the highest rating game using a loop.

Evaluation criteria: You should have updated in the file game.js

- ❑ The function getHighestRating to get the highest rating game using a loop
- ❑ Only 1 if statement is needed, inside the loop

3.2.3 Exercise: Show your favourite games using arrays

We still have one function that is using our hard-coded game objects instead of the "games" array: printFavouriteGames. We should update this function to use a loop as well.

Update the printFavouriteGames function to get loop over all the games and print the ones that have the *favourite* property set to *true*.

Evaluation criteria: You should have updated in the file game.js

- ❑ The function printFavouriteGames to print all games where the *favourite* property is set to *true*
- ❑ Only use 1 conditional, inside the.

3.3 Array destructuring

3.3.1 Syntax

We have now updated all our functions to use our new “game” array. We now also want to show the first 2 games in our array. We can extract the first 2 items from an array by using the destructuring syntax. In the case of our farm animals, this goes as follows:

```
const animals = [ /* All our animals */ ];
const [first, second] = animals;
```

3.3.2 Exercise: Destructuring the first 2 games from our "games" array

Extract the first 2 games from our array and show their name at the bottom of the page by calling addStatus for each of them. Add a meaningful title for them of course.

Evaluation criteria: You should have updated in the file game.js

- ❑ Added at the bottom some code to extract the first 2 games using the destructuring syntax.
- ❑ Added at the bottom some code to show these 2 games on our page.

You can now re-arrange the items in your "games" array to check that your page displays different results.

3.4 Add games of your best friend

3.4.1 Syntax

Your best friend sees that you are keeping track of your games and would like to use the same overview. You decide to merge your list of games with their games, but keep them in 2 separate arrays. When you want to print all your games, you can *spread* the 2 arrays together. When we want to spread 2 arrays together, we can do this as follows:

```
const animals = [ /* All our animals */ ];
const otherAnimals = [ /* Some other animals */ ];
const allAnimal = [...animals, ...otherAnimals]
```

3.4.2 Exercise: Add an array for games from your best friend

Update the printAllGames function with an array as parameter instead of using the “games” array by default. Also update the loop inside this function to use this received array. Update all invocations of the printAllGames function to include this new parameter. Also move the instruction to print a title from the printAllGames function outside of the function, and print it *before* calling the printAllGames function.
Add a new array, “friendGames”, and put some games inside it. Print these games at the bottom of your page by calling the printAllGames function using this new array.

Evaluation criteria: You should have updated in the file game.js

- ❑ Updated the printAllGames function to receive an array as parameter
- ❑ Updated the printAllGames to loop over the received array instead of the “games” array.
- ❑ Updated all invocations of printAllGames to pass the “games” array
- ❑ Removed the addStatus call that prints the title inside printAllGames to outside the function

- ❑ Added at the bottom a new array with games from your friend.
- ❑ Added at the bottom some code to show these games.

3.4.3 Exercise: Show all the games, both my own and the ones from my best friend

Add some code that will print all the games that we have, both my own games and the ones from my friend. Do this by spreading the 2 arrays together into 1 array and calling passing this new array on to the printAllGames function as a parameter

Evaluation criteria: You should have updated in the file game.js

- ❑ Created a new array at the bottom that contains both my games and the ones from my friend. Use the spread operator for this.
- ❑ Displayed these games using the printAllGames function.