P.PORTO

ESMAD | TSIW | POO
Exercise Sheet nº 8
Arrays

Open Visual Studio Code and solve the following exercises in independent files with the names **exN.html** and **exN.js**, where **N** is the number of the exercise:

1. Creation, access, transforming, counting, and iteration)

a. Create a **countries** array with the names of the countries Portugal, Spain and France
b. Print the first element in an alert box
c. Print the last element in an alert box
d. Add a new element: German
e. Change the German element to Germany
f. Print the following message in an alert box:

   The array has X countries!

   (where X indicates the number of current elements in the array)
g. Create another array called **countries2**
h. Equal (attribution operator) the created array to the **countries** array
i. Add a new element ("Denmark") to the **countries2** array
j. Print the **countries** array in an alert box. Analyze the results.
k. Iterate (using a for loop) over all elements of the array by displaying the country names on the console as follows:

   1 - Portugal

   2 - Spain

   …
l. Solve the previous exercise using the **for...of** cycle

2. Array methods - add/remove elements

a. Create an empty array **names**
b. Include a text box and an "ADD NAME" button on the web page. By pressing the button, you must add the contents of the text box to the array and print the contents of the array to the console
c. Add 5 names ("John", "Peter", "Mary", "Ann", "Michael")
d. Extract the last name from the array and display it in an alert box
e. Extract the first name of the array and display it in an alert box
f. Create a second array called names2 with the elements: "Charles", "Paul"
g. Join both arrays in a new array called **allNames** and show it on the console
h. Solve the previous question using spread syntax

i. Improve point (b) so it shouldn't be possible to insert repeated names

3. Array methods - search for elements

a. Create a numeric array called **grades** with OOP student grades (4,12,16,11,8)
b. Print the first positive note of the array in an alert box
c. Create a **getPositiveGrades** function that should return an array with all the positive notes. Then print an alert box with the contents of that array.
d. Create a **checkDisapprovals** function that should return a logical value indicating whether negative notes exist or not. Print the result on the console.
e. Create a **validateGrades** function that should return a logical value indicating whether the notes in the array are valid. Valid grades are all grades between 0 and 20 inclusive. Print the result on the console.

4. Array methods - transform arrays

a. Create a numeric array called **ages** with the age of alDisplay game information on the consolel members of a family (1, 9, 41, 75, 44, 73)
b. Create a new array called **fiveYearsFromNowAges** with the family ages 5 years from now. Print the array in an alert box
c. Print the sum of all ages on the console
d. Print the average of all ages on the console
e. Create a new array called **adults** with ages under 18 transformed into "-" and the rest unchanged
f. Order the age array and present the result in an alert box
g. Invert the result of the previous paragraph and save it in a new array called **inverseOrderedAges**

5. Array & Objects

a. Create the next object literal:

```
const grades = [
  { name: 'Peter', grade: 8 },
  { name: 'Ann', grade: 12 },
  { name: 'Mary', grade: 14 }
]
```

b. Create a function that creates a new object (with user data) and insert them into the array
c. Create a function that calculates the average of the OOP grades
d. Create function that lists the names of students with a positive grade

6. Array & Objects

a. Create the next array:

```
const games = [
  {  name: 'Rick Dangerous', year: 1989 },
  { name: 'Goblins', year: 1992 },
  { name: 'Golden Axe', year: 1989 }
]
```

b. Show in the console information about the games:

Name: N / Year: A

…

(where N is the name of the game and A is the year the game was released)

c. Display the number of existing games on the console

d. Create a function called **addGame** that creates a new object (with user data from 2 text boxes and an "ADD GAME" button) and insert them into the array

e. Improve the previous paragraph so as not to let an existing game be inserted. If it already exists, an alert box should be displayed with the text: "GAME ALREADY EXISTING!"

f. Create a function called **averageLaunchDate** that calculates and returns the average year of game launch. Display the result in an alert box

g. Create a new button called "REMOVE GAME" which, when pressed, must remove the game whose name was entered in the text box. If the game does not exist, the message: "GAME INEXISTENT" must be displayed in an alert box

h. Create a function called **gamesFrom90** that returns the names of games released in the 90s. Display the result in an alert box

i. Create a function called **concatenateGameNames** that returns a string with all game names in lowercase and without spaces. Display the result in an alert box

7. Arrays & Objects

Make a web page that allows you to manage your favorite movies. The page must consist of a form at the top with the following fields:

- Film title (required)
- Genre (adventure, action, comedy, horror, romance, thriller, Other)
- Year (between 1900 and the current / required)
- Cover (link / required)
- Trailer (link)

- Buttons:
    - Submit
    - Reset

When adding a valid movie (note: you cannot add two movies with the same title) a table with the following columns must be fed:

- Title
- Genre
- Options (set of buttons)
    - SEE COVER (opens modal with the movie cover)
    - SEE TRAILER (opens modal with automatic trailer playback)
    - REMOVE (removes the movie from the object array)