Maps, sets, métodos de array, iteradores… y otros

Título	Descripción	Enlace
Maps	Una introducción a los maps y a los weakmaps. **Con leer por encima suficiente.**	https://lenguajejs.com/javascript/set-map/que-es-map-weakmap/
Sets	Una introducción a los sets y a los weaksets. **Con leer por encima suficiente.**	https://lenguajejs.com/javascript/set-map/que-es-set-weakset
Expresiones regulares	Una introducción a las expresiones regulares**. Con leer por encima, suficiente.**	https://www.w3schools.com/js/js_regexp.asp
Fechas y horarios	Una introducción a los Datetimes.	https://www.w3schools.com/js/js_dates.asp
Iterables	Una introducción a los objetos iterables. **Lectura diagonal.** Lo más importante son el punto del Symbol.iterator y el resumen.	https://es.javascript.info/iterable
Generadores	Introducción a los generadores. **Lectura diagonal**. El punto más interesante es el de funciones generadoras y el resumen.	https://es.javascript.info/generators
Modulos	OPCIONAL. Es una lectura densa, pero puede ser interesante a futuro que os suene el tema de las importaciones y exportaciones y las diferentes maneras.	https://es.javascript.info/modules

EJERCICIOS

Maps-sets

1. Crea una función que sea capaz de coger dos sets y ver qué valores están en ambos.
2. Crea una función que sea capaz de coger dos sets y juntarlos en un tercer set, ¿habrá valores repetidos?
3. Crea una función que saque un set con los valores diferentes entre dos sets dados.
4. Crea una función para ver si un set es un subset de otro.
5. Crea una función para encontrar el valor máximo y el valor mínimo de un set
6. Crea una función para obtener todos los valores de un set que estén entre dos rangos (numéricos)

7. Crea una función que printee por consola todos los pares key value de un map.
8. Haz todos los ejercicios de la página: https://www.w3resource.com/javascript-exercises/javascript-regexp-exercises.php

1. Write a JavaScript program to test the first character of a string is uppercase or not.

2. Write a JavaScript program to check a credit card number.

3. Write a pattern that matches e-mail addresses.
The personal information part contains the following ASCII characters.

- Uppercase (A-Z) and lowercase (a-z) English letters.
- Digits (0-9).
- Characters ! # $ % & ' * + - / = ? ^ _ ` { | } ~
- Character . (period, dot or fullstop) provided that it is not the first or last character and it will not come one after the other.

4. Write a JavaScript program to search a date within a string.

5. Write a JavaScript program that work as a trim function (string) using regular expression.

6. Write a JavaScript program to count number of words in string.
Note :
- Remove white-space from start and end position.
- Convert 2 or more spaces to 1.
- Exclude newline with a start spacing.

7. Write a JavaScript function to check whether a given value is IP value or not.

8. Write a JavaScript function to count the number of vowels in a given string.
Test Data :
console.log(alphabetize_string('United States'));
Output :
"SUadeeinsttt"

9. Write a JavaScript function to check whether a given value is an valid url or not.

10. Write a JavaScript function to check whether a given value is alpha numeric or not.

11. Write a JavaScript function to check whether a given value is time string or not.

12. Write a JavaScript function to check whether a given value is US zip code or not.

13. Write a JavaScript function to check whether a given value is UK Post Code or not.

14. Write a JavaScript function to check whether a given value is Canada Post Code or not.

15. Write a JavaScript function to check whether a given value is a social security number or not.

9. Si quieres seguir con las expresiones regulares: https://learnbyexample.github.io/learn_js_regexp/Exercise_solutions.html
10. Haz todos los ejercicios de fechas de esta página: https://www.w3resource.com/javascript-exercises/javascript-date-exercises.php

1. Write a JavaScript function to check whether an `input` is a date object or not. Go to the editor

Test Data :
console.log(is_date("October 13, 2014 11:13:00"));
console.log(is_date(new Date(86400000)));
console.log(is_date(new Date(99,5,24,11,33,30,0)));
console.log(is_date([1, 2, 4, 0]));
Output :
false

2. Write a JavaScript function to get the current date.

Note : Pass a separator as an argument.
Test Data :

```
console.log(curday('/'));
console.log(curday('-'));
```
Output :
"11/13/2014"
"11-13-2014"

3. Write a JavaScript function to get the number of days in a month.

Test Data :
```
console.log(getDaysInMonth(1, 2012));
console.log(getDaysInMonth(2, 2012));
console.log(getDaysInMonth(9, 2012));
console.log(getDaysInMonth(12, 2012));
```
Output :
31
29
30
31

4. Write a JavaScript function to get the month name from a particular date.

Test Data :
```
console.log(month_name(new Date("10/11/2009")));
console.log(month_name(new Date("11/13/2014")));
```
Output :
"October"
"November"

5. Write a JavaScript function to compare dates (i.e. greater than, less than or equal to).

Test Data :
```
console.log(compare_dates(new Date('11/14/2013 00:00'), new Date('11/14/2013 00:00')));
console.log(compare_dates(new Date('11/14/2013 00:01'), new Date('11/14/2013 00:00')));
console.log(compare_dates(new Date('11/14/2013 00:00'), new Date('11/14/2013 00:01')));
```
Output :

"Date1 = Date2"
"Date1 > Date2"
"Date2 > Date1"

6. Write a JavaScript function to add specified minutes to a Date object.

Test Data :
console.log(add_minutes(new Date(2014,10,2), 30).toString());
Output :
"Sun Nov 02 2014 00:30:00 GMT+0530 (India Standard Time)"

Write a JavaScript function to test whether a date is a weekend.

Note : Use standard Saturday/Sunday definition of a weekend.
Test Data :
console.log(is_weekend('Nov 15, 2014'));
console.log(is_weekend('Nov 16, 2014'));
console.log(is_weekend('Nov 17, 2014'));
Output :
"weekend"
"weekend"
undefined

8. Write a JavaScript function to get difference between two dates in days.

Test Data :
console.log(date_diff_indays('04/02/2014', '11/04/2014'));
console.log(date_diff_indays('12/02/2014', '11/04/2014'));
Output :
216
-28

9. Write a JavaScript function to get the last day of a month.

Test Data :
console.log(lastday(2014,0));
console.log(lastday(2014,1));
console.log(lastday(2014,11));
Output :

31
28
31

10. Write a JavaScript function to calculate 'yesterday day'.

Test Data :
console.log(yesterday('Nov 15, 2014'));
console.log(yesterday('Nov 16, 2015'));
console.log(yesterday('Nov 17, 2016'));
Output :
"Fri Nov 14 2014 00:00:00 GMT+0530 (India Standard Time)"
"Sun Nov 15 2015 00:00:00 GMT+0530 (India Standard Time)"
"Wed Nov 16 2016 00:00:00 GMT+0530 (India Standard Time)"

11. Write a JavaScript function to get the maximum date from an array of dates.

Test Data :
console.log(max_date(['2015/02/01', '2015/02/02', '2015/01/03']));
Output :
"2015/02/02"

12. Write a JavaScript function to get the minimum date from an array of dates.

Test Data :
console.log(min_date(['2015/02/01', '2015/02/02', '2015/01/03']));
Output :
"2015/01/03"

13. Write a JavaScript function that will return the number of minutes in hours and minutes.

Test Data :
console.log(timeConvert(200));
Output :
"200 minutes = 3 hour(s) and 20 minute(s)."

14. Write a JavaScript function to get the amount of days of a year.

Test Data :
console.log(days_of_a_year(2015));
365
console.log(days_of_a_year(2016));
366

15. Write a JavaScript function to get the quarter (1 to 4) of the year.

Test Data :
console.log(quarter_of_the_year(new Date(2015, 1, 21)));
2
console.log(quarter_of_the_year(new Date(2015, 10, 18)));
4

11. Para practicar con módulos, vete a esta página web y ejecuta el ejercicio completo: https://education.launchcode.org/intro-to-professional-web-dev/appendices/exercise-solutions/modules.html#modules-solutions3

Iteradores-generadores-otros

12. Crea una función generadora de Fibonacci
13. Pasa eslint y prettier por todos tus scripts y mira qué cosas y fallas te salen.
14. Crea un generador del producto de los elementos de un array
15. Crea un generador que devuelva el máximo, el mínimo valor de un objeto iterable (array: [2, 4, 3, 5] -> (2, 2) -> (2,4) -> (2, 4) -> (2, 5)
16. Crea un iterador que devuelva infinitamente el valor de un número más la suma de un valor step cada vez. (valor:10, step:2 -> 10, 12, 14, 16....)
17. Crea un iterador infinito que devuelva todas las keys de un objeto y vuelva a empezar
18. Crea tres excepciones diferentes y un código que las utilice en diferentes puntos.
19. Crea dos excepciones y un código que controle dichas excepciones en diferentes puntos.

20. Crea un iterador de números naturales
21. Crea un iterador de números primos (Ojo!)
22. Crea un iterador de números pares
23. Crea un iterador de números múltiples de 3 impares
24. Intenta crear en un iterador las siguientes sucesiones:
    i. 1, 0, 0, 0, 0, 0, 1, 0, 2, 1, 1, 0,...
    ii. 1, 2, 3, 5, 10, 19, 20, 30, 1000...
    iii. 6, 2, 5, 5, 4, 5, 6, 3, 7,...
    iv. 1, 4, 9, 61, 52, 63, 94...
25. Write a function called divide that takes two parameters: a numerator and a denominator. Your function should return the result of numerator / denominator. However, if denominator is zero you should throw the error, "Attempted to divide by zero."
26. Ejercicio complet de https://education.launchcode.org/intro-to-professional-web-dev/chapters/exceptions/exercises.html:

A teacher has created a `gradeLabs` function that verifies if student programming labs work. This function loops over an array of JavaScript objects that *should* contain a `student` property and `runLab` property.

The `runLab` property is expected to be a function containing the student's code. The `runLab` function is called and the result is compared to the expected result. If the result and expected result don't match, then the lab is considered a failure.

```
function gradeLabs(labs) {
   for (let i=0; i < labs.length; i++) {
      let lab = labs[i];
      let result = lab.runLab(3);
      console.log(`${lab.student} code worked: ${result === 27}`);
   }
 }

let studentLabs = [
   {
      student: 'Carly',
      runLab: function (num) {
         return Math.pow(num, num);
      }
   },
   {
      student: 'Erica',
      runLab: function (num) {
         return num * num;
      }
   }
];

gradeLabs(studentLabs);
```

The `gradeLabs` function works for the majority of cases. However, what happens if a student named their function incorrectly? Run `gradeLabs` and pass it `studentLabs2` as defined below.

```
let studentLabs2 = [
   {
      student: 'Blake',
      myCode: function (num) {
         return Math.pow(num, num);
      }
   },
   {
      student: 'Jessica',
      runLab: function (num) {
         return Math.pow(num, num);
      }
   },
   {
      student: 'Mya',
      runLab: function (num) {
         return num * num;
      }
   }
];

gradeLabs(studentLabs2);
```

Upon running the second example, the teacher gets `TypeError: lab.runLab is not a function`.

Add a `try/catch` block inside of `gradeLabs` to catch an exception if the `runLab` property is not defined. If the exception is thrown, `result` should be set to the text `"Error thrown"`.

27. Create a function called safe_int() that takes a single argument i. If possible, the function converts i to int and returns it. If not possible (i.e. if an Exception occurs), the function returns None.
28. Define a function capitalize_last_name() that accepts as argument a string with a (single) first and a (single) last name, and returns a string in which only the first letter of the first name is uppercase, whereas all letters of the last name are uppercase; in otherwords, 'marisa tomei' becomes 'Marisa TOMEI'. (Tip: use str.split() to split a str into separate words.) If something other than a str object is passed as an argument, the function should raise a TypeError. (Tip: you can use isistance() to check whether an object is of a particular type.) If the str does not consist of exactly two words, the function should raise a ValueError.

29. Localiza el error en el siguiente bloque de código. Crea una excepción para evitar que el programa se bloquee y además explica en un mensaje al usuario la causa y/o solución:

```
lista = [1, 2, 3, 4, 5]
lista[10]
```

30. Realiza una función llamada **agregar_una_vez(lista, el)** que reciba una lista y un elemento. La función debe añadir el elemento al final de la lista con la condición de no repetir ningún elemento. Además si este elemento ya se encuentra en la lista se debe invocar un error de tipo *ValueError* que debes capturar y mostrar un mensaje en su lugar

BIBLIOGRAFÍA

¿Qué es un Map? - Javascript en español. (s. f.). Lenguaje JS. https://lenguajejs.com/javascript/set-map/que-es-map-weakmap/

¿Qué es un Set? - Javascript en español. (s. f.). Lenguaje JS. https://lenguajejs.com/javascript/set-map/que-es-set-weakset/

W3Schools.com. (s. f.). https://www.w3schools.com/js/js_regexp.asp

W3Schools.com. (s. f.-b). https://www.w3schools.com/js/js_dates.asp

Kantor, I. (s. f.). *Módulos, introducción*. https://es.javascript.info/modules-intro

https://www.w3resource.com/javascript-exercises/javascript-regexp-exercises.php, 2022-07-21

https://learnbyexample.github.io/learn_js_regexp/Exercise_solutions.html, 2022-07-21

https://education.launchcode.org/intro-to-professional-web-dev/appendices/exercise-solutions/modules.html#modules-solutions3, 2022-07-21

El tutorial de JavaScript Moderno. (s. f.). https://es.javascript.info/