Programación con JavaScript II

Sesión sincrónica 3

Nuevos tipos y características

Bienvenida y actividad de bienestar

Duración: 10 minutos.

Nombre de la práctica: La mejor versión de ti mismo.
Descripción de la práctica: Escribe acerca de la mejor versión posible de ti mismo durante al menos 20 minutos.
Palabras clave: Emociones positivas, fortalezas de carácter, autorregulación y esperanza.

Instrucciones para el aprendedor:
Imagina que dentro de 20 años has crecido en todas las áreas o maneras que te gustaría crecer y las cosas te han salido tan bien como te las imaginaste.

- ¿Cómo es esa mejor versión de ti mismo?
- ¿Qué hace él o ella cotidianamente?
- ¿Qué dicen los demás acerca de él o ella?

No es necesario que compartas este escrito, ya que el objetivo de esta reflexión es enfocarse en la experiencia que viviste mientras reflexionabas en esa mejor versión posible de ti mismo.

Fuente: Ejercicio contribuido por Taylor Kreiss de University of Pennsylvania Positive Psychology Center, y basado en el libro A Primer in Positive Psychology de Christopher Peterson.

Actividad guiada Parte 1

Duración: 75 minutos.

En este tema pondrás en práctica algunas de las nuevas características y tipos de datos aprendidos durante la lección. Para alcanzar el objetivo, realiza las siguientes actividades:

Ejercicio 1

Crea una función que, dado un arreglo de elementos, elimine los duplicados. Por ejemplo: MiFuncion([“10”,”X”,”X”,”2”,”10”,10,true, true]) devolverá [“10”,”X”,2,10,true]. Valida que el arreglo se haya ingresado, que el valor ingresado sea un arreglo, que el arreglo no esté vacío y que el arreglo tenga al menos dos elementos.

1. Crea un documento html en el directorio que hayas destinado para tus archivos de la sesión guiada 2 del curso Programación con JavaScript II.
2. En ese mismo directorio, crea una carpeta en donde guardarás los archivos js.
3. En esta carpeta, crea el archivo nuevo.js.
4. Enlaza el archivo js al archivo index.html.
5. Crea la función flecha **quitarDuplicados** y pásale como parámetro default la variable arr, asignándole el valor **indefinido.**
6. Valida que el valor de la variable **arr** no este indefinida, en caso de estarlo, envía un warning a la consola indicando el texto “no ingresaste un arreglo”.
7. Para validar si el valor ingresado es un arreglo, utiliza **instanceofArray.** En caso de que no sea un arreglo, envía un error a la consola indicando la cadena “El valor que ingresaste no es un arreglo”.
8. Para validar que el parámetro ingresado no este vacío, utiliza el método **length** del parámetro. En caso de ser igual a 0, coloca en la consola un error que indique “El arreglo está vacío”.
9. Para validar si el arreglo tiene al menos dos valores, utiliza nuevamente el método **length** del parámetro, pero validando que el tamaño del arreglo sea igual a 1.
10. Regresa a la consola con un info un objeto con dos miembros:
    a. El primero con la llave original del cual su valor es el arreglo que pasó como parámetro.
    b. El segundo miembro del objeto tiene la llave **sinDuplicados** y el valor utiliza el método filter al arreglo original y pásale como parámetro una función flecha que reciba como parámetros el **valor**, el **índice** y **self.** La función flecha lo que hará es aplicar el método **indexOf(valor=== índice)**.

Actividad guiada Parte 1

11. Prueba tu código con los siguientes casos de prueba:
    a. **quitarDuplicados**();
    b. **quitarDuplicados**({});
    c. **quitarDuplicados**([]);
    d. **quitarDuplicados**([2]);
    e. **quitarDuplicados**([“10”,”X”,”X”,”2”,”10”,10,true, true]);
12. Mejora el filtrado del arreglo utilizando programación funcional y el objeto Set. Para ello, haz lo siguiente:
    a. Comenta el código desarrollado en el punto 10.
    b. Regresa a la consola con un info un objeto con dos miembros:
        i. El primero con la llave original del cual su valor es el arreglo que pasó como parámetro.
        ii. El segundo miembro del objeto tiene la llave sinDuplicados y un arreglo formado por [... new Set(arr)].
    c. Prueba nuevamente tu código con los casos de prueba del punto 11.

Ejercicio 2

En este ejercicio vas a utilizar el tipo de dato **map.**
Crea una función que devuelva un arreglo limpio de anagramas. Crea dos versiones de la función, una utilizando un objeto plano y otra utilizando Map.

Un anagrama es un conjunto de palabras que tienen el mismo número de letras, pero en orden distinto. Por ejemplo:

roma – amor – mora

sopa – paso – sapo

ropa – paro – proa

- Utiliza el mismo html creado en el archivo anterior, solo apunta al nuevo js que se va a crear.
- En la carpeta js, crea el archivo map.js.

Actividad guiada
Parte 1

Visión con objeto plano:

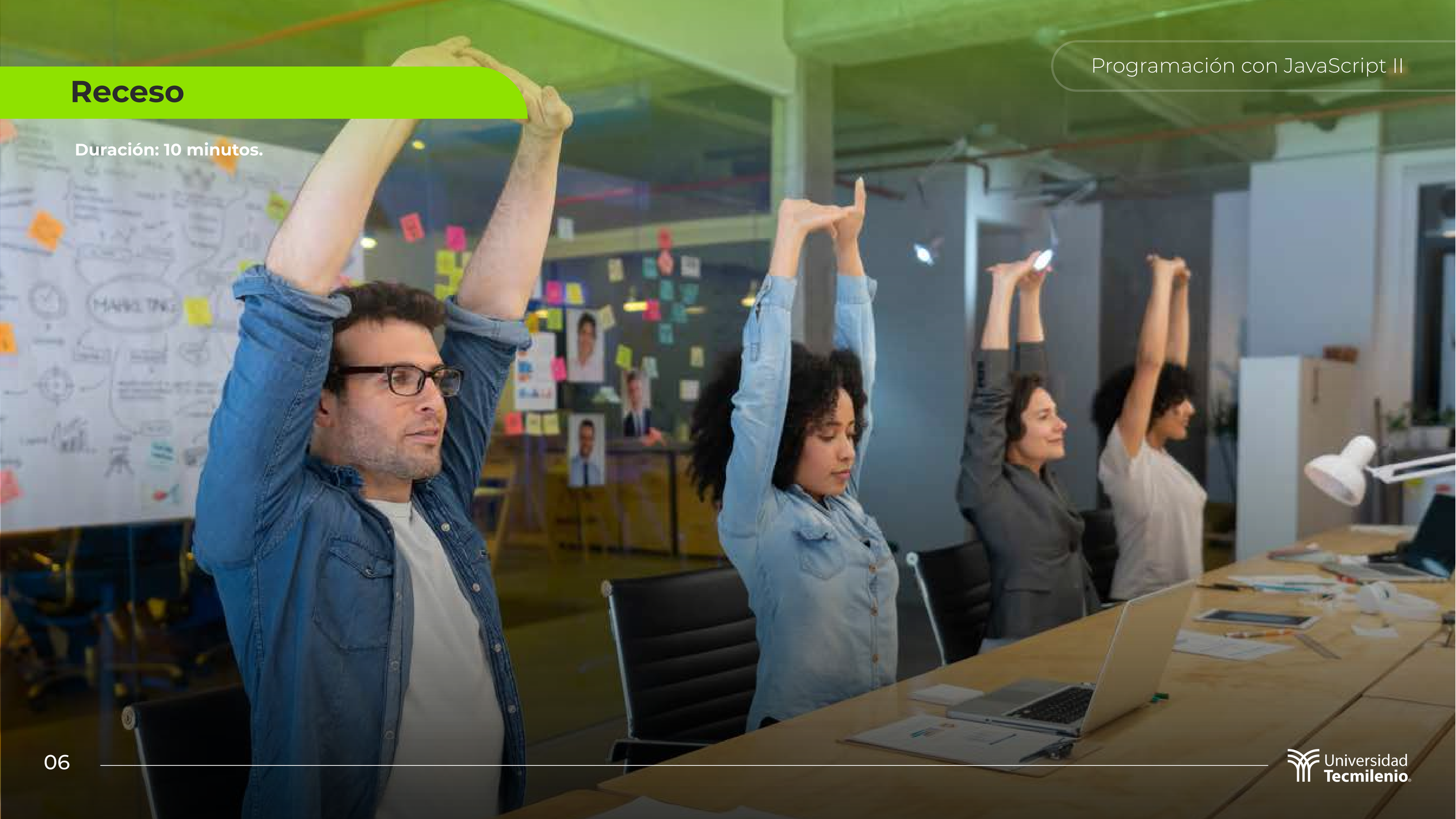
1. Dentro del archivo map, crea la función **limpiaAnagrama**, la cual recibirá como parámetro el arreglo.
2. Declara un objeto vacío.
3. Declara un for que va a iterar sobre el arreglo pasado como parámetro:
   a. Declara la variable **ordenado** e iguálala al elemento del arreglo al que le vas a aplicar los métodos **toLowerCase**(), **split**(), **sort**(), **join**() para dividir la palabra por letras, ordenarlas y volverlas a unir.
   b. Asigna al elemento del objeto iterado el valor de arreglo iterado de la siguiente manera: obj[sorted]=arr[i].
4. Con un return, utiliza el método **values** de la clase **Object** a la cual le pasarás como parámetro el objeto obj.
5. Declara el arreglo que se quiere evaluar.
6. En un **alert**, manda a llamar la función **limpiaAnagrama**(arreglo).

Versión con tipo de dato Map:

1. Declara la variable **limpiaAnagramaMap**, la cual recibe como parámetro un arreglo.
2. Declara la variable mapa, que va a ser igual a un nuevo **Map**().
3. Crea un for of:
   a. Evalúa cada palabra (variable declarada en el for) del arreglo pasado como parámetro.
   b. Declara la variable **ordenado** e iguálala al elemento del arreglo a la que le vas a aplicar los métodos toLowerCase(), split(), sort(), join() para dividir la palabra por letras, ordenarlas y volverlas a unir.
   c. Utiliza la función set de mapa para fijar **ordenado y palabra** como **clave, valor**.
4. Con un return, utiliza el método **from** de la clase **Array** para pasar como parámetro los valores de mapa.
5. Declara el **arreglo** que se quiere evaluar.
6. En un alert, manda a llamar la función **limpiaAnagramaMapa**(arreglo).

Receso

Duración: 10 minutos.



Actividad guiada Parte 2

Duración: 75 minutos.

Ejercicio 3

Retomando la función que genera el cuadrado de un número, modifícala para que puedas hacerlo utilizando generadores e iteradores.

1. Utiliza el mismo html creado en el archivo anterior, solo apunta al nuevo js que se va a crear.
2. En la carpeta js, crea el archivo **generadores.js.**
3. En el archivo, crea la función **cuadrado** y pasa como parámetro un valor.
4. Dentro del cuerpo de la consola, ejecuta un setTimeout que recibirá como parámetro una función anónima, lo único que hará es regresar la impresión en consola (un objeto con el valor y el resultado) y como temporizador utiliza un número aleatorio multiplicado por 1000.
5. Regresa el objeto formado por **valor** y **resultado.**
6. Crea una función generadora llamada **generador** sin parámetros.
7. Dentro del cuerpo de la función, imprime el texto “Inicia Generador”.
8. Luego utiliza un **yield** pasándole como parámetro 0, otro pasándole 1 y así sucesivamente hasta 5.
9. Imprime en consola el texto “Termina Generador”.
10. Declara la función **gen** en la que asignarás la llamada a la función **generador.**
11. Crea un for/of en el que recorras la variable **gen** para cada **yield** e imprimas en consola cada elemento **yield**.

Ejercicio 4

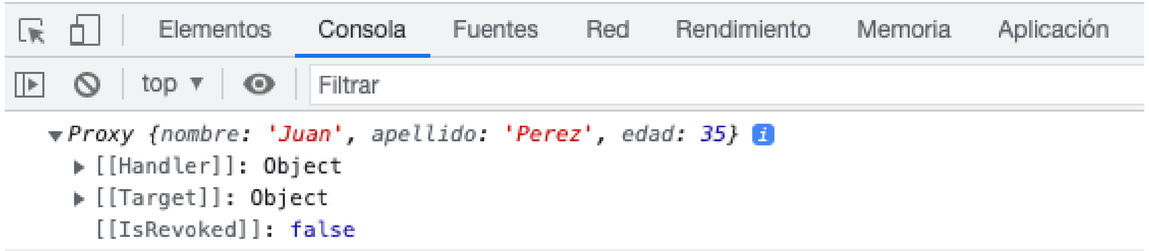
Utiliza un proxy para la validación de los valores de propiedades de un objeto **persona** que tiene tres propiedades: **nombre, apellido** y **edad.**

1. Utiliza el mismo html creado en el archivo anterior, solo apunta al nuevo js que se va a crear.
2. En la carpeta js, crea el archivo **proxy.js.**
3. Crea el objeto base llamado **persona** con las propiedades **nombre**, **apellido** y **edad**. Establece los valores **nombre** y **persona** como vacíos y el valor de **edad** como 0.
4. Crea el **manejador**, el cual es un **objeto** que recibe un parámetro set, que, a su vez, recibe dentro tres parámetros: el objeto, las **propiedades** que se desean evaluar (nombre, apellido y edad) y el **valor**. Además, al **objeto** en cuestión, en cada una de las tres propiedades modificadas, asígnale el valor que el usuario está proporcionando.

Actividad guiada
Parte 2

Ejercicio 4

5. Realiza las siguientes validaciones:
   a. Crea un nuevo proxy y asígnale un nombre. Dicho proxy recibirá como parámetros los objetos **persona** y **manejador**.
   b. Asigna valores a la propiedad **nombre**, **apellido** y **edad**.
6. Envía a la consola el proxy creado.
7. Asignando los valores nombre= “Juan”, apellido= “Pérez”, edad=35, tu salida en este momento tendría que ser algo parecido a esto:



8. Ahora crea una propiedad juan.twitter=@juanperez.
9. Observa el resultado en la consola.
10. Imprime en la consola el objeto **persona**. ¿Qué ves? ¿Por qué tiene esos valores?
11. Dentro del set del **manejador**, evalúa que exista en la lista del objeto original una propiedad, esto lo puedes realizar con el método indexOf. En caso de que regrese un –1, retorna por medio de un error en la consola “La propiedad ${propiedad_en_cuestion} no existe en el objeto persona”.
12. Por ejemplo, toma el caso de Twitter:
    a. Observa el resultado en tu navegador.
    b. Crea una nueva validación antigüedad que valide el **nombre** y **apellido** y que solo acepte letras y espacios, puedes realizarlo utilizando la siguiente expresión regular:
    (/^[A-Za-zÑñÁáÉéÍíÓóÚúÜü\s]+$/g.test(valor))

Actividad guiada Parte 2

Consideraciones:

Instructor: Hay algunos temas sobre las nuevas funciones que posiblemente el usuario no conozca, procura poner énfasis para detectar y explicar esos temas.

Cierre

Duración: 10 minutos.

Con esta serie de ejercicios, has puesto en práctica algunas de las principales características que implementó ECMAScript a partir de su versión 6. Es importante que las domines para que puedas escribir código más eficiente y moderno. Sin embargo, ten en cuenta que algunas de las nuevas funcionalidades no se ejecutan correctamente en todos los navegadores, pero no te preocupes, en los próximos temas se solucionará ese pequeño detalle.