

Programación con Java Script I

Sesión sincrónica 6

Programación funcional



Bienvenida y actividad de bienestar

Duración: 10 minutos.

Nombre de la práctica: Comunicarse positivamente

Descripción de la práctica: Practicarás la retroalimentación activa y constructiva para mejorar la interacción con las personas.

Palabras clave: Emociones positivas, resiliencia, perspectiva

Instrucciones para el participante: Martin Seligman señala que existen cuatro formas de abordar la comunicación con otra persona:

1. Actividad destructiva Señalar aspectos negativos de un evento o una conversación	4. Actividad constructiva Apoyo auténtico y con entusiasmo
2. Pasiva destructiva ignorar el evento o la conversación	3. Pasiva constructiva Apoyo breve, sin seguimiento o por compromiso

Seligman señala que es sumamente importante cultivar la retroalimentación activa constructiva, ya que esta ayuda a que tu interlocutor experimente emociones positivas y se concentre en sus fortalezas, no en sus debilidades. ¡Practícalo!

1. Conscientemente busca dar retroalimentación activa y constructiva a las personas con las que hables durante un día.
2. Observa cómo se comportan luego de recibir una respuesta de tal tipo. Llena los resultados de tus conversaciones en una tabla.
3. Reflexiona, ¿habrían tenido el mismo comportamiento si hubieras abordado las conversaciones con un enfoque distinto?

Fuente: Seligman, M. (2011). Building Resilience. Recuperado de <https://hbr.org/2011/04/building-resilience>

Actividad guiada

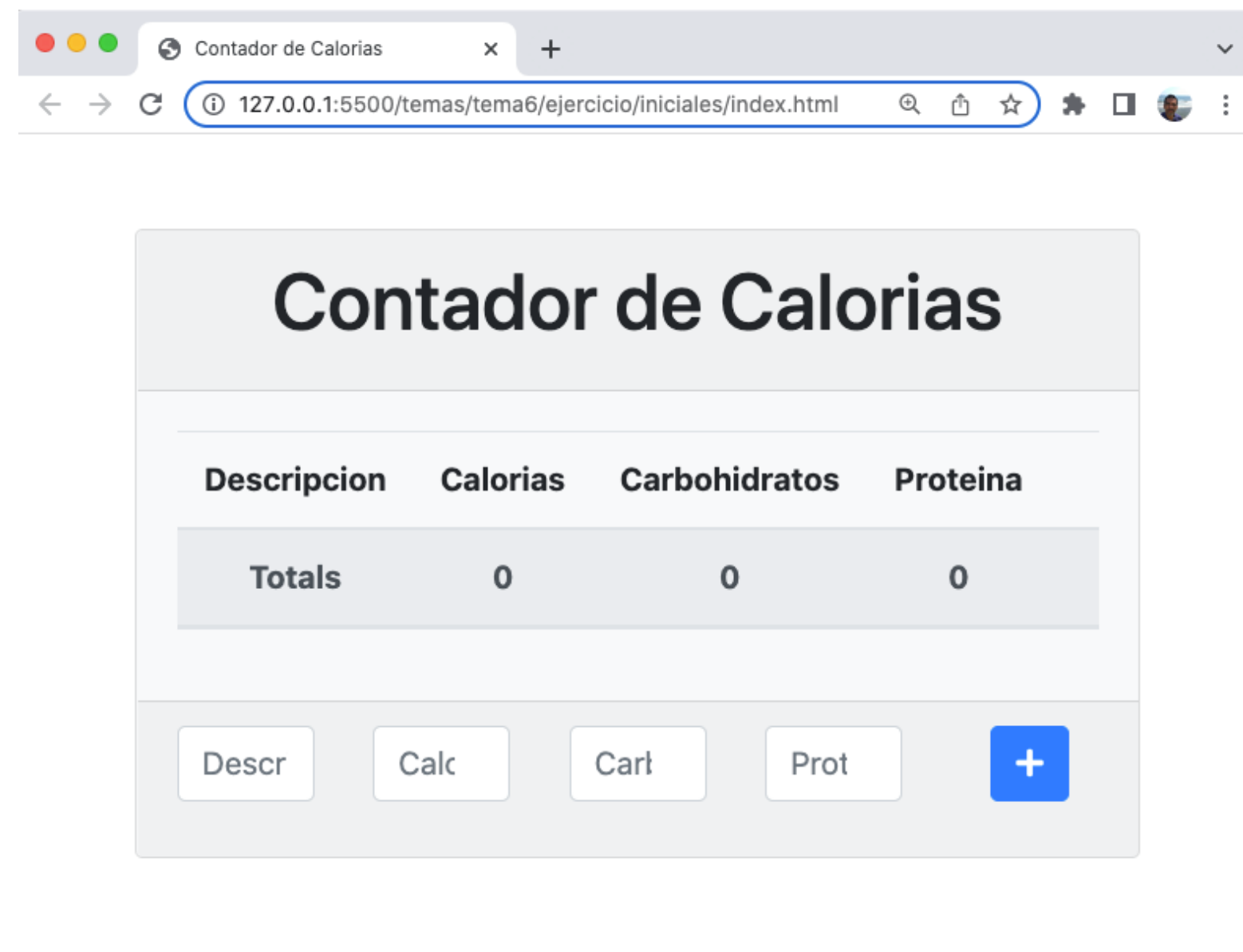
Parte 1

Duración: 75 minutos.

En el siguiente ejercicio vas a practicar la programación funcional aprendida en esta experiencia educativa, a través de la creación de una aplicación de conteo de calorías, carbohidratos y proteínas de cada alimento que ingrese el usuario.

Para alcanzar el objetivo, realiza las siguientes actividades:

1. Descarga la carpeta **ejercicio.zip** y descomprímela.
2. Coloca tu carpeta ejercicio en tu área de trabajo y ábrela con el VSC.
3. Revisa el código del archivo html y js para que te familiarices con los componentes, ejecuta tu código y podrás validar que deberá tener el siguiente aspecto:



Actividad guiada

Parte 1

El objetivo es que cuando el usuario introduzca valores en **descripción, calorías, carbohidratos y proteínas**, estos se agreguen a una lista en la que va a ser desplegada la información de cada uno de los elementos agregados y se tendrá la suma de todos los elementos. Además, se deberán poder agregar a través de un botón y también se deberán poder eliminar los registros de forma individual. Al final del proyecto, la página deberá lucir de la siguiente manera con los elementos agregados:

Contador de Calorias				
Descripción	Calorías	Carbohidratos	Proteína	
Leche	100	150	200	
Manzana	200	50	400	
Carne	50	80	500	
Totales	350	280	1100	
Descripción	Calorias	Carbohid	Proteina	

Comienza validando que el usuario ingrese un valor en un campo y que el tipo de dato corresponda. Para esto es necesario que se validen cada uno de los campos, lo puedes hacer obteniendo los valores del archivo html. Para poder hacerlo con jQuery:

1. Asigna un id a cada input en el archivo html. Te tendría que quedar un código parecido a esta imagen:

Actividad guiada

Parte 1

```
<div class="card-footer">
  <div class="row mb-2">
    <div class="col">
      <input type="text" class="form-control mb-2 mr-sm-2" id="descripcion"
        placeholder="Descripcion">
    </div>
    <div class="col">
      <input type="number" min="0" class="form-control mb-2 mr-sm-2" id="calorias"
        placeholder="Calorias">
    </div>
    <div class="col">
      <input type="number" min="0" class="form-control mb-2 mr-sm-2" id="carbohidratos"
        placeholder="Carbohidratos">
    </div>
    <div class="col">
      <input type="number" min="0" class="form-control mb-2 mr-sm-2" id="proteina"
        placeholder="Proteina">
    </div>
    <div class="col">
      <button class="btn btn-primary mb-2">
        <i class="fas fa-plus"></i>
      </button>
    </div>
  </div>
</div>
```

2. Crea cuatro variables en el archivo **main.js** para acceder a los inputs, una por cada input (**descripcion, calorias, carbohidratos y proteina**)
3. Crea una variable de tipo arreglo en la que se van a almacenar los elementos una vez que se cumplan con las validaciones.
4. En el archivo **index.html**, crea un evento **onclick** en el botón que agrega elementos a fin de poder validar si el usuario ingresó algún valor. Haz que el botón valide a través de la función **validarInput()**.

Actividad guiada

Parte 1

```
<div class="col">
<button class="btn btn-primary mb-2" onclick="validarInput()">
<i class="fas fa-plus"></i>
</button>
</div>
```

5. En el archivo **main.js**, crea esta función expresada y valida que los campos input no estén vacíos utilizando el método **val()** de cada input. En caso de que estén vacíos, agrega la **clase de bootstrap 'is-invalid'**, para lo cual puedes utilizar un operador ternario similar a este para cada elemento:

```
descripcion.val() ? "" : descripcion.addClass('is-invalid');
```

6. El resultado de esto tendría que visualizarse así:

Contador de Calorias			
Descripción	Calorías	Carbohidratos	Proteína
Totales	0	0	0
Descripcior ✖	Calorias ✖	Carbohid ✖	Proteina ✖
<div>+</div>			

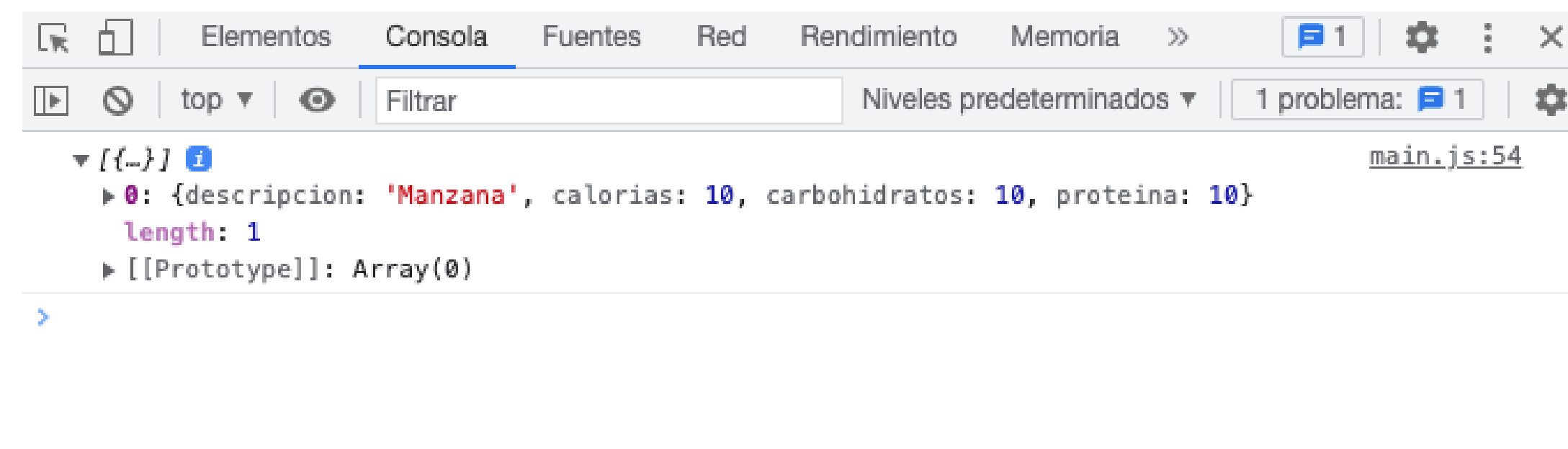
Actividad guiada

Parte 1

7. Si se coloca un valor, es necesario eliminar la clase **'is-invalid'**. Para lograrlo, utiliza el método **.keypress()** de jQuery, manda llamar a este método en un código que deberás colocar por fuera de la función **validarInput()** (de preferencia que lo coloques antes de la función), pasándole una función flecha con el método **removeClass** a cada elemento. Por ejemplo:

```
carbohidratos.keypress(() => {  
  carbohidratos.removeClass('is-invalid');  
})
```

8. Regresando a la función **validarInput()**, valida que todos los inputs tengan un valor, si eso ocurre, entonces agrega ese elemento a la lista mandando a llamar a la función **agregar()**.
9. Como aún no existe la función **agregar**, créala debajo de **validarInput()** de la siguiente manera:
- Declárala como función expresada.
 - Genera un nuevo objeto, que es el que se va a agregar a la lista, nómbralo **nuevoElemento** y llénalo con los valores del input.
 - Asegúrate de que los valores para calorías, carbohidratos y proteínas sean numéricos, esto lo podrías lograr utilizando `parseInt`.
 - Con el objeto ya completo, utiliza el método **push** para que lo agregues a la lista.
 - Para probar, podrías imprimir en consola tu lista y debería arrojar algo parecido a esto:



Actividad guiada

Parte 1

10. Cada que se introduce un elemento, los inputs no se limpian, por lo que es necesario limpiarlos cada que se ingresa un elemento a la lista. Para realizarlo, crea una función **limpiarInputs** en la que le vas a pasar un valor vacío a cada input de la siguiente manera:

```
descripcion.val("");
```

11. Realízalo para cada elemento y manda a llamar a esta función desde la función agregar, de tal manera que una vez que se agregue un elemento, inmediatamente ponga los valores de los inputs en blanco.

Ahora se van a agregar los elementos en el **tbody** dinámicamente con funciones de composición. Se sugiere que, para dar orden al archivo, se creen estas funciones justo después de la función **compose** que ya venía incluida en el archivo.

1. Crea una función **attrsToCadena**:
 - a. Debe recibir dos parámetros: el tag y otro objeto, que son las clases de este tag.
 - b. Obtén las llaves del objeto que se está pasando, antes crea una variable y utiliza la función de JavaScript **Objet.keys** del objeto que se está pasando.
 - c. Crea otra variable **atributos** del tipo array.
 - d. Crea un ciclo for con el que vas a recorrer el contenido de **keys**.
 - e. Guarda en una nueva variable atributo el atributo que va a ser igual a la llave de la posición y este se va a agregar al arreglo que se creó mediante un **push** de la manera atributo = atributo del objeto.
 - f. Genera el string utilizando la función de JavaScript **.join** que regresa el contenido convertido en una cadena.
 - g. Regresa el string resultante.

Actividad guiada

Parte 1

2. Crea la función **attrsTag** que reciba un objeto y retorne la cadena completa para el html:
 - a. A esta función se le van a pasar dos parámetros: un objeto que contiene el tag y la clase con sus atributos. Además, se requiere pasarle el contenido de ese tag.
 - b. Debe validar si la parte en donde vienen la clase y sus atributos no está vacía, en caso de estarlo, solo regresará el tag sin clases que modifiquen su apariencia o comportamiento. Por otro lado, en caso de que sí traiga atributos, mandará a llamar a la clase **attrsToCadena** y esta función convertirá esos atributos en cadena que se pintará después del tag, luego el contenido y el tag de cerradura.
 - c. Debe retornar un string.
3. Crea una función expresada que armará los tags:
 - a. Nombra a esta función **tag**.
 - b. Esta función valida si el tipo de dato del objeto recibido es un string, si lo es, entonces manda a llamar a la función **attrsTag** (atributos de tag) y se le pasará como atributo la propiedad tag (tag:t). En caso contrario, solo pásale tag (t).
4. Crea una función a la que nombrarás **celdaTabla**, esta va a generar una sola celda y la forma de hacerlo es utilizando la función tag. Pásale como argumento la cadena 'td'.
5. Como no se desea llamar a esa función cada vez que se va a hacer una celda, construye una función que lo haga por ti:
 - a. Nómbrala **celdasTabla**, la cual recibirá un arreglo. Para obtener cada uno de los elementos de ese arreglo, utiliza la función map de JavaScript, ejecuta celdaTabla y unifica con la función .join().
6. Ahora, crea una función para armar una fila, nómbrala **filaTablaTag** y esta será igual a la invocación de la función tag con la cadena 'tr', que es el tag que forma una fila en una tabla.
7. Crea una función que arme las filas con sus respectivas celdas, la cual deberá llamarse **filaTabla** y recibirá un arreglo de items. Es aquí donde utilizarás la función compose, a la cual le pasarás como argumentos la función **filaTablaTag** y celdasTabla, pasando también los items aparte de la siguiente manera:

```
const filaTabla = elementos => compose(filaTablaTag, celdasTabla)(elementos);
```

Receso

Duración: 10 minutos.

```
def __init__(self, **kwargs):
    self.name = kwargs.get("name")
    self.damage = kwargs.get("damage")
    self.armor = kwargs.get("armor")
    self.hit_points = kwargs.get("hp")
    self.current_hit_points = kwargs.get("hp")
    self.level = kwargs.get("level")

def attack(self, enemy: 'Unit') -> int:
    """
    Attack enemy unit. Return number of damage
    """
    damage_top_limit = self.damage + random.randint(0, 10)
    damage_bot_limit = self.damage - random.randint(0, 10)
    calculated_damage = random.randint(damage_bot_limit, damage_top_limit)
    if calculated_damage < 0:
        return 0
    enemy.current_hit_points -= calculated_damage
```

Actividad guiada Parte 2

Duración: 75 minutos.

Ahora vamos a crear la función que va a contabilizar y actualizar los totales de cada categoría e imprimirlos en su respectivo lugar.

1. Crea una función flecha **actualizarTotales**, la cual va a recorrer un arreglo a través de **.map**:
 - a. Declara tres variables: **calorias**, **carbohidratos** y **proteina** e inicialízalas en 0.
 - b. Utiliza la función **map** para recorrer el arreglo **lista** y suma el contenido de cada uno de los elementos **calorias**, **carbohidratos** y **proteina** a su respectiva variable.
 - c. Para poder mostrar los totales, utiliza la función **.text** de jQuery y asígnala al selector de esta manera:

```
$('#totalCalorias').text(calorias);
```

-
-
-
- d. Realízalo de esta manera para los tres selectores. Para poder visualizar los resultados, invoca a la función **actualizarTotales** () desde la función **agregar()**.
- e. Ejecuta nuevamente tu aplicación y se debería visualizar más o menos así:

Contador de Calorias			
Descripción	Calorías	Carbohidratos	Proteína
Totales	13	13	13

Descr

Calc

Carl

Prot

+

Actividad guiada

Parte 2

2. Para pintar los elementos, utiliza la función **append** de JavaScript, creando la función **renderElementos**:
 - a. Asegúrate de que la lista esté vacía. Para ello, utiliza el selector **tbody** y el método **.empty** para vaciarlo.
 - b. Recorre la lista con el método **.map**.
 - c. Dentro del **.map**, utilizando el método **.append** de jQuery, llama a la función **filaTablaTags** y pásale como argumento los elementos del array para que construya las etiquetas que se insertarán en el cuerpo de la tabla.
 - d. Para que puedas visualizar los resultados, invoca a la función **renderElementos** desde la función **agregar**.
 - e. En este momento te debe comenzar a mostrar los elementos que se van agregando y debería de tener una apariencia similar a la siguiente:





Contador de Calorias			
Descripción	Calorías	Carbohidratos	Proteína
manzana	12	12	12
pera	15	15	15
babana	58	12	65
Totales	85	39	92

Actividad guiada

Parte 2

Ahora, crea la funcionalidad para eliminar un elemento:

1. Primero crea un ícono que se va a utilizar como botón:
 - a. Declara el ícono como **iconoBorrar** y llama a la función **tag** para que construya una etiqueta 'i' y que le ponga las clases fa y fa-trash-alt de Bootstrap. Pásale solo una cadena vacía al segundo argumento (coloca este código en la “zona de definición” para que tengas tu código más ordenado).
 - b. Modifica la función renderElementos para poder crear el botón y hazlo de la siguiente manera:
 - i. Además de elemento, pásale como argumento el índice del elemento al map.
 - ii. Crea una función botonEliminar para que se pinte en el html invocando a la función tag.
 1. Pásale como primer atributo los parámetros el tag ' button' y:
 - a. class: 'btn btn-outline-danger'
 - b. onclick: `eliminarElemento(\${indice})`
 2. Como segundo atributo, pásale el iconoBorrar.
 - iii. Agrega botonEliminar a la invocación de la función filaTabla.
2. Después de realizar estos cambios, tu aplicación debe lucir así:

Contador de Calorias				
Descripción	Calorías	Carbohidratos	Proteína	
manzana	12	12	12	
pera	15	15	15	
banana	45	45	45	
Totales	72	72	72	
Descr	Calc	Carb	Prot	

Actividad guiada

Parte 2

Ahora falta crear la función para eliminar elementos:

1. Crea la función **eliminarElemento**:

- Utiliza la opción **splice** de JavaScript, la cual requiere que le pases como parámetro el índice del elemento que se desea eliminar y el número de elementos (en este caso siempre será 1).
- Llama a la función **actualizarTotales()** para que se actualicen los totales cuando se elimine el elemento.
- Llama a la función **renderElementos()** para que se redibujen en el html.

Tu ampliación ya está completa, solamente es prudente que las funciones **atrsToCadena** y **tag** estén en su versión funcional y no imperativa:

- Cambia el for de la función **atrsToCadena** por un **.map**.
- Cambia el if de la función **tag** por un operador ternario.

Consideraciones

Instructor, refuerza los conocimientos de aquellos temas que consideres necesarios para que el aprendedor pueda realizar este ejercicio.

La obra presentada es propiedad de ENSEÑANZA E INVESTIGACIÓN SUPERIOR A.C. (UNIVERSIDAD TECMILENIO), protegida por la Ley Federal de Derecho de Autor; la alteración o deformación de una obra, así como su reproducción, exhibición o ejecución pública sin el consentimiento de su autor y titular de los derechos correspondientes es constitutivo de un delito tipificado en la Ley Federal de Derechos de Autor, así como en las Leyes Internacionales de Derecho de Autor.

El uso de imágenes, fragmentos de videos, fragmentos de eventos culturales, programas y demás material que sea objeto de protección de los derechos de autor, es exclusivamente para fines educativos e informativos, y cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por UNIVERSIDAD TECMILENIO.

Queda prohibido copiar, reproducir, distribuir, publicar, transmitir, difundir, o en cualquier modo explotar cualquier parte de esta obra sin la autorización previa por escrito de UNIVERSIDAD TECMILENIO. Sin embargo, usted podrá bajar material a su computadora personal para uso exclusivamente personal o educacional y no comercial limitado a una copia por página. No se podrá remover o alterar de la copia ninguna leyenda de Derechos de Autor o la que manifieste la autoría del material.