Ayuda para construir la aplicación

0. Preliminares

Antes de comenzar a escribir las funciones que manejarán los eventos es conveniente definir variables que permitan acceder a los botones, al elemento de input y al párrafo del resumen de activas e inactivas.

También necesitaremos una variable que permita acceder a la lista de tareas (ver el punto siguiente) y al elemento div desde el cual estarán colgados los elementos p de cada una de las tareas. Puede usar const o let dependiendo si espera que esa variable no cambie (por ejemplo los botones) o que cambie (listas de cosas)

Finalmente necesitamos una variable que lleve el id de la siguiente tarea a crear inicializado en 0. Cada ingreso de una nueva tarea debe incrementar este número

Luego de declarar esas variables es necesario adosar *listeners* a los botones y al elemento de input.

Por ejemplo: *botonActivas.AddEventListener('click', desplegarActivas)*

1. El Modelo de Datos

El modelo en esta aplicación corresponde a las tareas que están registradas en la aplicación y podemos simularlo como un simple array en que cada elemento es una tarea.

Cada tarea es un objeto con la siguiente estructura

```
task = {
    'id': 3,                    // un número entero que se incrementa
    'active': true,
    'what': "hacer el trabajo de JavaScript"
}
```

Por ejemplo, en un instante dado podríamos tener en el arreglo de tareas lo siguiente:

```
tareas = [{'id':0, 'active':true, 'what':"estudiar JavaScript"},  {'id':1, 'active':false, 'what':"estudiar Rails"}]
```

Para recorrer todo el arreglo de tareas puede usar la estructura de control siguiente:

```
for t of tareas {
  // y aquí tratar cada tarea del arreglo
 // por ejemplo t['id'] devuelve el id, t{'active'] el valor true o false
}
```

2. Representación en el DOM de la lista de tareas

Puedes usar un elemento para colgar desde ese elemento las tareas

Cada tarea puede ser simplemente un elemento <p> que debe ser creado dinámicamente a partir de un objeto

t de la lista de tareas. Cada elemento p tiene que tener 3 atributos:

- un id que permita identificarlo y que lo haremos coincidir con el id del objeto tarea
- un atributo que indique si está activa o inactiva
- un atributo class que servirá para presenta de distinta forma los p activos de los p inactivos

Por ejemplo, en un instante dado tendremos una lista de tareas como:

```
tareas = [ {'id':3, 'active':true, 'what':"algo"}, {'id':5, 'active':false, 'what':"algo mas"},
            {'id':6, 'active':true, 'what':"otra cosa"}  ]
```

Lo que dará origen a lo siguiente:

```
<div id="laLista">
      <p id="3" active="true" class="activa"> algo </p>
      <p id="5" active="false" class="inactiva"> algo mas </p>
      <p id="6" active="true" class="activa"> otra cosa </p>
</div>
```

Cada uno de los elementos p tendrá ademas adosado un "event listener" atento a un click en el elemento para cambiarlo de activo a inactivo o de inactivo a activo

3. Creación de un nuevo elemento en la lista de tareas

- crear el elemento con e = document.createElement("p")
- agregar un listener para detectar un click en el (activar/desactivar)
    - e.addEventListener('click', manejarclick)
    (manejarclick() es una función que se encargará de cambiar esa tarea de activa en inactiva o viceversa)
- agregar los atributos del elemento p (id, active, class)
    - crear primero el atributo, por ejemplo att = document.createAttribute('id')
    - darle valor al atributo, por ejemplo att.value = t.id
    - agregar el atributo al nuevo elemento por ejemplo e.setAttributeNode(att)
- finalmente agregar el nuevo elemento p a la lista l.appendChild(e)

4. Una función clave a implementar es algo como lo siguiente:

function agregarAlista(laLista, laTarea) que hace todo lo se detalla en el punto 3.
Una vez que disponemos de esta función generar la lista de tareas en el dom es simplemente

```
for (laTarea of tareas) {agregarAlista(laLista, laTarea)
```

Para obtener las activas es muy similar

```
for (laTarea of tareas) {
  if(laTarea.active) {
     agregarAlista(laLista, laTarea)
  }
}
```

5. Manejo de los clicks en los párrafos

Otra función clave es la función manejarclick(event) que se encarga de manejar el click sobre un elemento p

- primero obtener el id del elemento p del click: let id = event.srcElement.id
- luego recorrer el arreglo de tareas para buscar cual corresponde al p que recibió el click (t.id == id)
- al encontrarlo hacer t['active'] = !t['active'] (si estaba activa inactiva y viceversa)

6. Línea Resumen

Necesitarás una función resumen() que genere la línea con la contabilidad de tareas totales, activas e inactivas. Para ello hay que recorrer el arreglo de tareas e ir contando las activas. Las totales estará dado simplemente por tareas.length e inactivas por la diferencia entre ambos valores.

7. Eliminar las Inactivas

Para eliminar las inactivas la función que debes escribir debe
- primero recorrer el arreglo de tareas e ir metiendo solo las tareas activas en un arreglo auxiliar (temp.push(t))
- vaciar el arreglo tareas haciendo tareas = []
- recorrer el arreglo auxiliar y volverlas a meter en tareas

8. La magia del CSS

¿Como lograr que las tareas inactivas aparezcan tachadas y de otro color?
Recuerda que cuando creamos los elementos p en forma dinámica le incorporamos 3 atributos uno de los cuales es class. Simplemente asignar a la clase del p correspondiente la forma en que queremos presentarlo. Por ejemplo, si las clases son activo e inactivo

```
p.inactiva    {color: darkred; text-decoration-line: line-through;}
p.activa      {color: navy; text-decoration:underline;}
```

9. Otros

- Es probable que debas llamar a la función que despliega todas las tareas no solamente cuando se haga un click sobre todas sino por ejemplo al ingresar o al cambiar de estado una tarea.

- Si quieres que tu aplicación parta con algunas tareas debe inicializar la lista de tareas. Por ejemplo, en la mía tengo:
  let tasks = [{'id':0, 'active':true, 'what':"estudiar JavaScript"}, {'id':1, 'active':true, 'what':"instalar Rails"}];
  let nextId = 2;

- Si quieres que la aplicación despliegue la lista de tareas al cargar puede agregar un atributo onload y una función que desea se ejecute al cargar
  Por ejemplo: <body onload="listAll()">