**Proyecto Integrador**

El proyecto integrador es un trabajo práctico que vas a ir desarrollando en el transcurso de las clases, en donde irás aplicando los conocimientos que vayas adquiriendo.

Vamos a desarrollar un **ToDo List**, que es una herramienta de productividad. Básicamente es un listado fácil e intuitivo de las tareas que tenemos pendientes y las que ya hemos realizado.

El proyecto lo vamos a ir desarrollando sobre la plataforma **Thimble** (https://thimbleprojects.org/egonzalez01/688880), y ya vas a contar con el diseño (HTML y CSS).

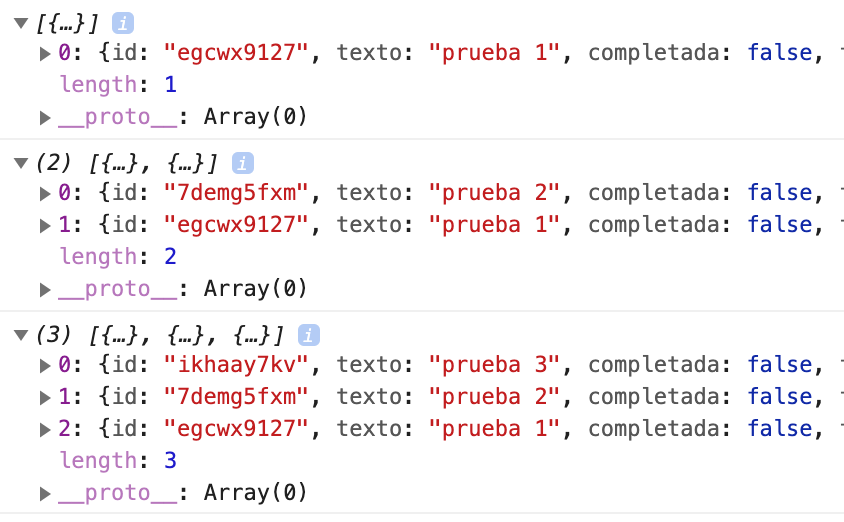
En esta clase, vamos a empezar a crear las principales variables que vamos a necesitar para trabajar, y algunas funciones específicas. Todo el código que vayamos desarrollando, lo vamos a ir guardando en el archivo **main.js**.

Es importante que utilicemos únicamente **const** o **let**. No podemos utilizar **var** para la creación de variables.

* En el archivo vas a tener ya creada una función llamada **nuevoID**. Es importante no borrar ni modificar esta función. Esta función la vamos a utilizar para crear un string conteniendo un valor único, el cual vamos a poder utilizar como identificador unívoco de cada tarea (algo así como un DNI para cada tarea).
* Además de la función, vas a tener dos contantes que tampoco debemos eliminar o modificar: **iconoEliminar** y **iconoCompletar**. Cada variable es la imagen a mostrar en el HTML de los íconos para cada acción que vamos a poder hacer sobre una tarea.
* Declará la variable **tareas**, y asignale un array vacío. Además de ir variando el contenido de la misma (ya que vamos a ir agregando y eliminando elementos), es importante tener en cuenta que necesitamos reasignarla.
* Dentro del HTML, tenemos 2 listados **<ul>**: uno de las tareas pendientes (con id ***“todo”***), y otro de las tareas completadas (con id ***“completed”***).  
  Utilizando getElementById o querySelector, vamos a obtener cada uno de los listados y guardarlos en dos constantes nuevas llamadas **tareasCompletadasEl** y **tareasPendientesEl**, respectivamente.
* Vamos a crear una función, utilizando **arrow functions**, llamada **crearTarea**, que tiene un único parámetro llamado ***texto*** (es el texto de la tarea que estamos creando), y va a retornar un objeto literal con las siguientes propiedades:
  + id: con el valor que retornar ejecutar la función **nuevoID** que ya estaba creada
  + texto: con el valor que llegó en el parámetro ***texto***. (como esta propiedad y el parámetro de la función tienen el mismo nombre, tenemos que utilizar **Object Property Value Shorthand**).
  + completada: una propiedad booleana que inicialmente va a tener el valor **false**
  + fechaCreacion: una propiedad que va a tener la fecha del momento en el que se crea el objeto
* Para probar que nuestra función está bien, podés agregar un código donde la ejecutamos, guardamos su valor de retorno en una variable y la mostramos por consola. Una vez que nos aseguramos que funciona bien, podemos borrar ese código.  
    
  **const nuevaTarea = crearTarea('prueba 1');**

**console.log( nuevaTarea );**

**// {id: "jzmkb280a", texto: "prueba 1", completada: false, fechaCreacion: Wed May 17 2019 21:33:52 GMT-0300 (Argentina Standard Time)}**

* Como siguiente paso, vamos a agregar un evento al botón para agregar una nueva tarea. Este botón tiene el id **add**. Por eso, lo primero que vamos a hacer es obtener ese botón utilizando **getElementById**, y agregar una función al evento **click**.
* Dentro de la función del evento click, vamos a tener por el momento 4 tareas:
  + Obtener el valor que escribimos en el input. Para esto, obtenemos por id “**item**” el elemento input y con **destructuring** guardamos la propiedad **value**.
  + Llamamos a nuestra función **crearTarea** y le pasamos como parámetro el **value** que obtuvimos el input. Guardamos el objeto que retorna esta función en una nueva variable.
  + Usando **spread operator**, agregamos la nueva tarea al principio del array **tareas** (acá vas a necesitar volver a reasignar el array tareas con el nuevo array resultado de agregar el elemento utilizando spread operator).
  + Mostrá por consola la variable **tareas**.
* Si todo funciona bien hasta acá, cada vez que escribas algo en el input y aprietes el botón **+**, vas a ver en consola un array con cada tarea nueva.  
    
  

Si llegamos hasta acá, ¡ya tenemos gran parte de nuestra herramienta de gestión de tareas desarrollada!

Ahora vamos a trabajar en hacer que cada vez que agreguemos una tarea, la misma se vea en el HTML, y no por consola como estamos haciendo hasta ahora.

Para lograr esto, vamos a realizar lo siguiente:

* Vamos a crear una **arrow function**, llamada **mostrarTareas** (sin parametros).
* Dentro de la función, lo primero que vamos a hacer es separar las tareas que tienen la propiedad **completada** en true (las tareas que ya completamos/finalizamos) y las que la tienen en false (las tareas que todavía están pendientes). Para esto, vamos a utilizar el método de los arrays **filter**. Las tareas que están finalizadas las vamos a guardar en una variable llamada c**ompletadas**, y las tareas que todavía estén sin finalizar las vamos a guardar en la variable **pendientes**.
* Cada tarea, en el HTML, tiene el siguiente formato (tanto las pendientes como finalizadas):  
    
  <li id="mnqwtwkam">

prueba 1

<div class="buttons">

<button class="remove">iconoEliminar</button>

<button class="complete">iconoCompletar</button>

</div>

</li>

El atributo **id** de <li> es la propiedad id del objeto tarea.

“**prueba 1”** es el texto del objeto tarea.

**iconoEliminar**/**iconoCompletar** deberían ser en realidad el valor que tiene cada variable, que ya están creadas al principio del código (“*<svg> …. </svg>*”).

* Utilizando el método **map** de los arrays, vamos a convertir cada lista de tareas en los elementos de HTML respetando la estructura anterior. Para crear esa estructura de una forma sencilla, vamos a utilizar **string literals**. De esta forma, podemos escribirla en múltiples líneas y hacer interpolación de textos para agregar los datos específicos de cada tarea.  
  El resultado del **map** sobre las tareas completadas lo vamos a guardar en la variable **completadasHTML**.  
  El resultado del **map** sobre las tareas pendientes lo vamos a guardar en la variable **pendientesHTML**.
* Una vez que tenemos ya los elementos a agregar en el DOM, simplemente vamos a modificar el contenido de ambas listas utilizando el siguiente código:  
    
  **tareasCompletadasEl.innerHTML = completadasHTML.join('');**

**tareasPendientesEl.innerHTML = pendientesHTML.join('');**

* Con estos puntos, ya tenemos la función **mostrarTareas** terminada. Ahora tenemos que ejecutarla dentro de la función que agregamos en el evento onclick del botón **add**, al final de todo (después de haber sumado la tarea al array).
* Si todo está bien, podemos refrescar nuestra web, y cada vez que agreguemos una tarea haciendo click en el botón **+**, se debería agregar un elemento nuevo en el listado de arriba (tareas pendientes).  
    
  