MEDAC
Instituto Oficial de Formación Profesional

Proyecto de JavaScript y DOM

Cristina Ahumada Martínez

TEMA Y FUNCIONAMIENTO DEL PROYECTO

El proyecto consiste en una página donde escribir tareas, con tres apartados:

- Una barra de texto con un botón de añadir para poder escribir la nueva tarea.
- Un apartado de tareas pendientes, donde se pondría tareas no realizadas. Tiene un contador donde aparecen la cantidad de tareas que hay en este bloque.
- Un apartado de tareas realizadas, donde irán las tareas pendientes que se realicen. Luego, de este apartado, pueden borrarse.

El funcionamiento, por tanto, sería: el usuario escribe la tarea dentro del cuadro de texto y le da al botón de Añadir. Aparece una ventana donde deberá introducir el título de la tarea, de forma que a la hora de ir al listado de tareas, solo aparezca este título.

Si el usuario quiere ver la descripción completa de la tarea deberá darle al botón de 'Mostrar' que aparece debajo del título. Así, la tarea no tiene por qué ocupar demasiado espacio y puede desplegarse u ocultarse a placer.

Si el usuario le da a Añadir pero no ha introducido nada en la tarea, entonces aparecerá una ventana emergente avisando de que no puede crear una tarea vacía.

Cuando el usuario haga realizado la tarea puede darle a la X roja, y esa tarea pasará automáticamente al bloque de Tareas realizadas. Queda así como una especie de inventario. Pero si la persona quiere borrar la tarea definitivamente y que no quede rastro de ella, puede darle a la X otra vez.

Cada tarea incluye la fecha de creación, y el número de caracteres.

DESCRIPCIÓN DE LOS EVENTOS UTILIZADOS

añadirTarea( ) → cuando se activa la función (siempre y cuando se haya introducido algo), se introduce el título en una ventana emergente y entonces aparecen los elementos que conforman el bloque de una tarea pendiente:

- un div que contendrá toda la información.
- un span donde estará el título de la tarea.
- un span donde estará la X para borrar la tarea.
- un párrafo donde aparece la tarea completa.
- un párrafo que tiene la palabra ocultar, que tendrá una funcionalidad.
- la fecha y el número de caracteres de la tarea.

Dentro de esta función también se hace una llamada a la función de eliminarTarea(), al que se le va a pasar el bloque con la tarea.

eliminarTarea( ) → esta tarea incluye dos resultados diferentes, dependiendo de dónde esté la tarea. Si la tarea que se quiere quitar está dentro del bloque de tareas pendientes, entonces se pasa la tarea al bloque de tareas terminadas. Pero si la tarea ya está dentro del bloque de tareas terminadas, entonces borrar directamente la tarea. Esto se consigue usando el parentNode.id.

Ocultar o mostrar la tarea → con el addEventListener he hecho que al pulsar sobre la palabra clave pueda expandirse o retraerse la tarea. Lo que he hecho ha sido poner un if y decirle que si el estilo de esa clave es 'none' (oculto), que cambie de estilo a otro donde sí podrá aparecer, y que el texto cambie. Y si no, pues que cambie a otro estilo diferente y que el texto sea Mostrar.

actualizarContador( ) → esta función modifica el elemento contador que creé en el html, que es un párrafo. Lo que he hecho ha sido guardar el elemento de 'tareas1' que es el bloque de tareas pendientes. Luego, con el getElementsByTagName he buscado todos los 'div' que hay dentro de ese bloque y se han guardado en una lista, y guardo esta lista en una variable. A continuación, con el .length averiguo cuántos elementos tiene esta variable y hago que el valor que tiene el párrafo (0 al inicio) cambie a la cantidad de divs -1.

Esta resta de un valor es porque al inicio he creado como una 'muestra' de la estructura pero no quería que apareciera realmente ningún texto al iniciar la página. Entonces, al crear una nueva tarea, el contador se ponía directamente en 2 porque contaba la tarea 'invisible' y la que había creado. Al restarle uno consigo ignorar esa tarea que no aparece.

Por último, he llamado a la función desde crear tarea y desde eliminar tarea. Así, cada vez que se cree una tarea y cada vez que se borre, la función contará cuantos divs hay y actualizará su valor.

cambiarEstilo( ) → en el css he creado una ristra de estilos asociados al modo-oscuro, donde hago cambios en general y también en bloques particulares como contenedor, button o tareas-pendientes. Luego, en js, he puesto que en la función se recoja la clase del cuerpo y he usado el toggle. El toggle me sirve para alternar entre clases: le paso una clase y si detecta que la tiene, la quita. Pero si detecta que no la tiene la pone.
Ya que en el html tengo puesta la función dos veces (cuando el ratón pasa por encima y cuando se quita), esta función se ejecutará dos veces, en una quitando el modo y en otra poniéndolo.

FUNCIONES DE TEXTO/ARRAYS EMPLEADAS

Fecha: he hecho que al crear la tarea aparezca la fecha actual. Para ello he guardado en una variable la fecha con `let f_actual = new Date();` , pero esta tendrá un formato especial. Lo siguiente, por tanto, ha sido pasarla a un formato específico, adaptado al español con `let f_texto = f_actual.toLocaleDateString('es-Es'),` y por último he insertado eso en la variable fecha con `fecha.innerHTML = f_texto + " | "`.

Caracteres: he querido introducir también una variable que muestre los caracteres de la tarea. Para ello he cogido el texto de la tarea y he recogido su longitud con `let n_carac = entrada.value.length`. Por último, lo he guardado la variable caracteres.

invertirTexto() → he usado una función para invertir ciertos textos, pero que depende de otra función. La función principal (la declarada en el html) es al revés(), y lo que hace es coger los id de las cosas que quiero cambiar y los guarda (como si fuera un array). Luego, con el forEach, voy cogiendo id a id y enviándolo a la función de invertirTexto().
La función de invertirTexto lo que hace es recibir los id y con el getElementById, coge el elemento (porque solo le he enviado el texto), y lo guarda en la variable elementos. Por último, cambio el texto de los elementos por otro nuevo, que será el que salga del split (separa por caracteres), reverse (los invierte), y join (los vuelve a juntar).

LO APRENDIDO DURANTE EL DESARROLLO

- El proceso de crear variables, guardarlas y luego manipularlas.
- A manejar mejor las etiquetas clave como "class" o "id".
- A diferenciar entre cuándo debo activar una función de forma simple o hacerlo con el addEventListener.
- Que puedo tener en el css estilos que no esté usando y cambiar de uno a otro según eventos determinados.
- Que en una variable puedo guardar directamente varios valores (como pasa en la función alreves()).