NUCBA

JAVASCRIPT

TO DO LIST

APUNTES

2.13 Livecoding: To Do List

En esta clase, realizaremos nuestro primer proyecto práctico del módulo de Javascript, y uno de los proyectos más comunes cuando estamos empezando con esta tecnología: **Una lista de tareas**.

Funcionalidades del proyecto

En este proyecto vamos a estar desarrollando las siguiente funcionalidades:

- Agregar una tarea a la lista mediante un formulario
- Validar si el input de tarea está vacío
- Validar si una tarea ya existe en la lista
- En caso de pasar las validaciones, creamos y renderizamos la tarea
- Guardaremos las tareas en el localStorage para persistir la lista
- Haremos aparecer/desaparecer el botón de borrar todas según corresponda
- Haremos la funcionalidad de borrar cada tarea de manera individual o todas juntas.

Objetivos del proyecto

Siendo este el primero proyecto práctico del módulo, el objetivo del proyecto es que puedan ir siguiendo paso a paso cómo se van realizando las distintas tareas del proyecto, para ver cómo se pueden manipular datos y convertirlos en elementos tangibles del árbol del DOM utilizando html y css de manera dinámica gracias a **Javascript**.

Además, este proyecto ayudará a entender cómo funciona el **localStorage** y cómo podemos trabajar con él y actualizarlo a medida que lo vayamos necesitando.

Es de vital importancia que no se queden con lo que vean en clase. Vuelvan a ver la realización del proyecto, saquense las dudas que tengan, pregunten, practiquen, intenten modificar cosas. Todo esto les va a permitir entender realmente cómo funciona el proyecto.

Dataset

Dataset en JavaScript es **un objeto que representa los atributos de datos personalizados que se pueden agregar a elementos HTML**. Este objeto se utiliza para **acceder y modificar** los valores de los atributos de datos personalizados (denominados "data attributes" en inglés) de un elemento HTML específico.

Los atributos de datos personalizados son un mecanismo de HTML que permite **agregar información adicional a un elemento HTML sin afectar a su representación visual o a su significado semántico**. Por ejemplo, se puede agregar un atributo de datos personalizado "data-product" a un elemento de lista para almacenar el identificador de un producto en un sitio de comercio electrónico.

Sintaxis

Dataset en JavaScript proporciona una forma conveniente de acceder a los valores de estos atributos de datos personalizados mediante una sintaxis sencilla y legible. Cada atributo de datos personalizado se representa como una propiedad en el objeto dataset, donde el nombre de la propiedad se forma a partir del nombre del atributo de datos personalizado, sin el prefijo "data-" y utilizando la convención camelCase. Por ejemplo, si un elemento HTML tiene el atributo de datos personalizado **"data-product-id"**, se puede acceder a su valor en JavaScript utilizando la propiedad **"productId**" del objeto dataset.

Veamos esto mismo en código

HTML:

```
<div id="product" data-product-id="12345" data-product-
name="Camisa" data-product-price="29.99"> </div>
```

JS:

```
// Obtener el elemento HTML que tiene atributos de datos personalizados
const product = document.querySelector('#product');

// Acceder a los valores de los atributos de datos personalizados utilizando Dataset
const productId = product.dataset.productId; // '12345'
const productName = product.dataset.productName; // 'Camisa'
const productPrice = product.dataset.productPrice; // '29.99'
```

Uso en el proyecto

Dataset en JavaScript es especialmente útil para manipular datos en aplicaciones web dinámicas y sitios web que utilizan HTML personalizado para almacenar información adicional.

En este proyecto, vamos a estar utilizando este objeto para almacenar los id’s generados para cada tarea en el botón de borrado, para que luego nos sea simple acceder al elemento del array de tareas que tenemos y borrarlo.

Date

En JavaScript, **Date** es un **objeto incorporado** que se utiliza para **trabajar con fechas y horarios**. Este objeto **proporciona una serie de métodos y propiedades** para crear, manipular y formatear fechas y horas en el formato de fecha y hora estándar del sistema.

La clase Date se puede utilizar para crear nuevos objetos de fecha. Por defecto, un objeto Date creado sin argumentos representa la fecha y hora actuales. Además, también es posible crear objetos Date con argumentos específicos para representar una fecha y hora específicas.

Sintaxis

Veamos algunos ejemplos de cómo podemos usar este objeto , con sus métodos más comunes.

```
// Crear un objeto Date que representa la fecha y hora actuales
const currentDate = new Date();

// Crear un objeto Date que representa una fecha y hora específicas
const specificDate = new Date('March 7, 2023 12:00:00');

// Obtener el año, mes, día, hora, minuto y segundo de un objeto Date
const year = specificDate.getFullYear(); // 2023
const month = specificDate.getMonth(); // 2 (los meses se indexan desde 0, por lo que marzo es 2)
const day = specificDate.getDate(); // 7
const hour = specificDate.getHours(); // 12
const minute = specificDate.getMinutes(); // 0
const second = specificDate.getSeconds(); // 0

// Formatear una fecha y hora en una cadena legible por humanos
const formattedDate = specificDate.toLocaleString(); // "March 7, 2023, 12:00:00 PM"
```

Uso en el proyecto : Date.now

No vamos a entrar en detalle acerca de las particularidades del uso del objeto Date en Javascript. Veremos únicamente el método **Date.now()** en este proyecto.

Date.now() es un método estático incorporado en la clase Date de JavaScript que **devuelve el número de milisegundos transcurridos desde el 1 de enero de 1970 00:00:00 UTC hasta la fecha y hora actuales**.

Ahora bien, ¿Por qué lo vamos a utilizar?

La respuesta es simple: Cada vez que generemos una nueva tarea, generaremos un id para la misma, para poder manipularla luego. Con este método, nos aseguramos de que el id de la tarea creada sea siempre distinto a las otras tareas, ya que los milisegundos transcurridos desde la fecha mencionada serán distintos cada vez que generemos una nueva tarea (Es imposible crear dos tareas al mismo tiempo).

Función inicializadora

En JavaScript, una función **init()** es una convención de nomenclatura comúnmente utilizada para nombrar una f**unción que se ejecuta al iniciar una aplicación o al cargar una página web**.

La función **init()** se puede utilizar para **realizar tareas de inicialización, como configurar el estado inicial de una aplicación, cargar datos de una fuente de datos o configurar los eventos y controladores de eventos necesarios para interactuar con la página**. Por lo general, la función init() se llama en la parte inferior del archivo JavaScript que se utiliza en una página web o en el archivo principal de la aplicación.

Uso en el proyecto

En nuestro proyecto, la vamos a estar utilizando para colocar todos los escuchadores de eventos, que queremos que estén disponibles desde que se carga la página.

Nucba tip: *Sigan investigando sobre los temas aquí expuestos. Practiquen, apoyense en la documentación oficial y sobre todo mantengan la constancia y la curiosidad a medida que vayan aprendiendo cosas nuevas.*

#HappyCoding 🚀