

Evaluación del módulo #4

Consigna del proyecto 

Proyecto: TaskFlow: Aplicación de Gestión de Tareas en JavaScript

Evaluación del módulo Programación avanzada en Javascript

Situación inicial

Unidad solicitante: Departamento de Desarrollo Web

El equipo de desarrollo ha recibido el encargo de construir una aplicación web interactiva basada en JavaScript. La problemática a resolver es ofrecer una herramienta funcional que permita a los usuarios gestionar tareas de manera eficiente. Se utilizará un enfoque basado en la orientación a objetos, eventos del DOM y consumo de APIs para lograr una aplicación moderna y escalable.

Nuestro objetivo

El objetivo de este proyecto es desarrollar una aplicación web interactiva que permita gestionar tareas de manera eficiente utilizando JavaScript moderno. Se implementarán principios de programación orientada a objetos, manipulación del DOM, eventos, asincronía y consumo de APIs para crear una herramienta escalable y funcional. La aplicación web debe permitir a los usuarios:

- Crear, editar y eliminar tareas.
- Utilizar eventos para mejorar la interactividad.
- Manejar datos de manera asincrónica.
- Integrar consumo de APIs para funcionalidades adicionales (como almacenamiento o sincronización de tareas).

Requerimientos

Requerimientos generales:

- Implementar principios de orientación a objetos en JavaScript.
- Usar sintaxis moderna de ES6+.
- Manipular el DOM de forma eficiente.
- Manejar eventos de usuario.
- Implementar funciones asincrónicas para trabajar con datos externos.
- Consumir una API para almacenamiento o funcionalidades extra.

Paso a paso

1. ORIENTACIÓN A OBJETOS EN JAVASCRIPT

Objetivo: Aplicar conceptos de POO para estructurar el código de la aplicación.

- Crear una clase Tarea con propiedades como id, descripcion, estado y fechaCreacion.
- Implementar métodos para cambiar el estado de la tarea y eliminarla.
- Crear una clase GestorTareas que administre una lista de tareas.

Entrega: Código con las clases implementadas y ejemplos de instanciación.

2. CARACTERÍSTICAS JAVASCRIPT ES6

Objetivo: Aplicar nuevas funcionalidades de JavaScript ES6+.

- Utilizar let y const en lugar de var.
- Aplicar template literals para mejorar la legibilidad del código.
- Usar arrow functions para mejorar la sintaxis de los métodos.
- Implementar destructuring y spread/rest operators en la manipulación de datos.

Entrega: Código refactorizado con sintaxis ES6 y ejemplos funcionales.

3. EVENTOS Y MANIPULACIÓN DEL DOM

Objetivo: Implementar eventos en la aplicación.

- Crear un formulario HTML para agregar nuevas tareas.
- Capturar eventos de submit y click para gestionar tareas.
- Modificar el DOM dinámicamente para mostrar la lista de tareas.
- Agregar eventos de mouseover y keyup para mejorar la interactividad.

Entrega: Aplicación funcional con eventos correctamente implementados.

4. JAVASCRIPT ASÍNCRONO

Objetivo: Implementar asincronía con setTimeout y setInterval.

- Simular un retardo al agregar una tarea.
- Implementar una función que muestre una notificación tras 2 segundos.
- Crear un contador regresivo para tareas con fecha límite.

Entrega: Código con funciones asincrónicas implementadas y ejemplos funcionales.

5. CONSUMO DE APIS CON JAVASCRIPT

Objetivo: Integrar una API externa para mejorar la aplicación.

- Usar `fetch()` para obtener datos de una API de tareas.
- Almacenar y recuperar tareas desde `localStorage`.
- Crear una función que guarde tareas en una API y otra que las recupere.
- Manejar errores en peticiones asíncronas con `try/catch`.

Entrega: Aplicación funcional con consumo de API y almacenamiento en `localStorage`.

¿Qué vamos a validar? 🔍

- Correcta aplicación de la orientación a objetos.
- Uso de ES6+ en la implementación del código.
- Interactividad lograda a través del manejo de eventos.
- Correcto manejo de asincronía.
- Implementación y validación del consumo de APIs.

Referencias 📖

- [MDN JavaScript](#)
- [Guía de ECMAScript 6](#)
- JSONPlaceholder API

Recursos 📁

- [POO en JavaScript - Artículo](#)
- [fetch\(\) y manejo de promesas - Documentación](#)
- [Ejemplo de aplicación en GitHub](#)

Entregables

- Código fuente documentado.
- Demostración funcional.
- Explicación del código en un informe breve.

Portafolio

Se recomienda subir este proyecto al portafolio personal en GitHub y LinkedIn, destacando la implementación de cada funcionalidad y los conocimientos adquiridos.

¡Éxitos!

Nos vemos más adelante

