Evaluación del módulo

Consigna del proyecto 👨‍💻

alkemy

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.

Resolución

```
// HTML para la aplicación
document.body.innerHTML = `
    <h1>TaskFlow: Gestor de Tareas</h1>
    <form id="formulario">
        <input type="text" id="tarea" placeholder="Agregar nueva tarea" required>
        <button type="submit">Agregar</button>
    </form>
    <ul id="lista-tareas"></ul>
`;

// Clase Tarea
class Tarea {
    constructor(id, descripcion) {
        this.id = id;
        this.descripcion = descripcion;
        this.estado = 'pendiente';
        this.fechaCreacion = new Date().toLocaleString();
    }
    cambiarEstado() {
        this.estado = this.estado === 'pendiente' ? 'completada' : 'pendiente';
    }
}
// Clase GestorTareas
class GestorTareas {
    constructor() {
        this.tareas = [];
    }
    agregarTarea(descripcion) {
        const id = Date.now();
        const nuevaTarea = new Tarea(id, descripcion);
        this.tareas.push(nuevaTarea);
        this.guardarEnLocalStorage();
    }
    eliminarTarea(id) {
        this.tareas = this.tareas.filter(tarea => tarea.id !== id);
        this.guardarEnLocalStorage();
    }
```

```
    cambiarEstadoTarea(id) {
        const tarea = this.tareas.find(tarea => tarea.id === id);
        if (tarea) tarea.cambiarEstado();
        this.guardarEnLocalStorage();
    }
    guardarEnLocalStorage() {
        localStorage.setItem('tareas', JSON.stringify(this.tareas));
    }
    cargarDesdeLocalStorage() {
        const datos = JSON.parse(localStorage.getItem('tareas')) || [];
        this.tareas = datos.map(t => new Tarea(t.id, t.descripcion));
    }
}

// Instancia del gestor y referencias del DOM
const gestor = new GestorTareas();
gestor.cargarDesdeLocalStorage();
const formulario = document.getElementById('formulario');
const listaTareas = document.getElementById('lista-tareas');

// Eventos
formulario.addEventListener('submit', (e) => {
    e.preventDefault();
    const input = document.getElementById('tarea');
    if (input.value.trim()) {
        gestor.agregarTarea(input.value);
        mostrarTareas();
        input.value = '';
    }
});
function mostrarTareas() {
    listaTareas.innerHTML = '';
    gestor.tareas.forEach(tarea => {
        const li = document.createElement('li');
        const span = document.createElement('span');
        span.textContent = `${tarea.descripcion} - ${tarea.estado}`;
        const btnCambiar = document.createElement('button');
        btnCambiar.textContent = 'Cambiar Estado';
        btnCambiar.addEventListener('click', () => cambiarEstado(tarea.id));
```

```
        const btnEliminar = document.createElement('button');
        btnEliminar.textContent = 'Eliminar';
        btnEliminar.addEventListener('click', () => eliminarTarea(tarea.id));
        li.appendChild(span);
        li.appendChild(btnCambiar);
        li.appendChild(btnEliminar);
        listaTareas.appendChild(li);
    });
}
function cambiarEstado(id) {
    gestor.cambiarEstadoTarea(id);
    mostrarTareas();
}
function eliminarTarea(id) {
    gestor.eliminarTarea(id);
    mostrarTareas();
}

// Asincronía
setTimeout(() => alert('Simulando retardo al agregar una tarea'), 2000);
setInterval(() => {
    console.log('Contador de tareas activado');
}, 5000);

// Consumo de API
async function obtenerTareasAPI() {
    try {
                                                const response = await
fetch('https://jsonplaceholder.typicode.com/todos?_limit=5');
        const data = await response.json();
        console.log('Datos obtenidos:', data);
    } catch (error) {
        console.error('Error al obtener datos:', error);
    }
}
obtenerTareasAPI();
mostrarTareas();
```

¡Éxitos!

Nos vemos más adelante

alkemy