INDICE

INTRODUCCIÓN

Ejercicio 1:

Diseña una estructura de datos que almacene la información de alumnos necesaria para la votación de delegado.

Esta estructura deberá ser un array de objetos alumnos, con las propiedades y métodos necesarios para la aplicación de votación de delegado.

Ejercicio 2:

Sobre el ejercicio de elección de delegado del aula, incorpora la estructura de datos diseñada en el ejercicio anterior para sustituir los arrays vinculados. Este ejercicio trabajará directamente con el array de objetos alumnos.

EJERCICIOS

Código:

```
class Alumno {
    constructor(id, nombreCompleto) {
        this.id = id;
        this.nombreCompleto = nombreCompleto;
    }
}

class Candidato extends Alumno {
    constructor(alumno) {
        super(alumno.id, alumno.nombreCompleto);
        this.votos = 0;
    }
}

let arrayDeAlumnos = [];
const buttonContainer = document.getElementById("buttons-container");
let candidatos = [];
let contador = 0;

function BotonesCandidatos() {
    let tamañoArray = arrayDeAlumnos.length;

    for (let i = 0; i < tamañoArray; i++) {
        const alumno = arrayDeAlumnos[i];
        const boton = document.createElement("button");
        boton.setAttribute("data-id", alumno.id);
        boton.textContent = alumno.nombreCompleto;
        buttonContainer.appendChild(boton);
        boton.addEventListener("click", seleccionCandidato);
    }
}
```

```
function cargarArray(fichero) {
    const urlArchivo = fichero;
    fetch(urlArchivo)
        .then (response => {
            if (!response.ok) {
                throw new Error("Eror al cargar el archivo. Codigo: " + response.status);
            }

            return response.text();
        })
        .then (contenido => {
            const lineas = contenido.split('\n');
            let idCounter = 1;

            arrayDeAlumnos = lineas
                .map(linea => linea.trim())
                .filter(linea => linea.length > 0)
                .map(nombre => new Alumno(`id-${idCounter++}`, nombre));

            console.log("Archivo cargado correctamente");
            console.log("Contenido de Array de Alumnos:", arrayDeAlumnos);

            BotonesCandidatos();
        })
        .catch (error => {
            console.error("Fallo durante la operacion de fetch o procesamiento:", error);

            const mensajeError = document.createElement("p");
            mensajeError.textContent = "Error: No se pudo cargar el archivo alumnos.txt.";
            mensajeError.style.color = "red";
            buttonContainer.appendChild(mensajeError);
        });
}

cargarArray("alumnos.txt")

function seleccionCandidato(event) {
    const elemento = event.currentTarget;
    const alumnoId = elemento.getAttribute("data-id");

    const yaSeleccionado = candidatos.some(c => c.id === alumnoId);

    if (!yaSeleccionado) {
        const alumnoBase = arrayDeAlumnos.find(a => a.id === alumnoId);
        if (alumnoBase) {
            const nuevoCandidato = new Candidato(alumnoBase);
            candidatos.push(nuevoCandidato);
            elemento.classList.add("seleccionado");
            console.log(`Candidato seleccionado: ${nuevoCandidato.nombreCompleto}`);
        }
    }
}
```

```
function enviarDelegados() {
    if (candidatos.length === 0) {
        const mensaje = document.getElementById("parrafo") || document.createElement("p");
        mensaje.setAttribute("id", "parrafo");
        mensaje.textContent = "Selecciona al menos un candidato antes de enviar.";
        document.getElementById("elegirCandidatos").insertAdjacentElement("afterend", mensaje);
        return;
    }

    buttonContainer.innerHTML = '';

    document.getElementById("enviarDelegados").style.display = "none";
    document.getElementById("elegirCandidatos").textContent = "Vota a un delegado";

    mostrarCandidatos();

    const padre = document.getElementById("elegirCandidatos");
    let p = document.getElementById("parrafo");

    if (!p) {
        p = document.createElement("p");
        p.setAttribute("id", "parrafo");
        padre.insertAdjacentElement("afterend", p);
    }

    p.textContent = arrayDeAlumnos[contador].nombreCompleto + " es tu turno para votar";
}

function mostrarCandidatos() {
    candidatos.forEach((candidato, index) => {
        const boton = document.createElement("button");
        boton.setAttribute("data-id", candidato.id);
        boton.textContent = candidato.nombreCompleto;
        buttonContainer.appendChild(boton);
        boton.addEventListener("click", votar);
    });
}
```

```
function votar(event) {
    if (contador >= arrayDeAlumnos.length) {
        mostrarResultados();
        return;
    }

    const elemento = event.currentTarget;
    const candidatoId = elemento.getAttribute("data-id");

    const candidatoVotado = candidatos.find(c => c.id === candidatoId);

    if (candidatoVotado) {
        candidatoVotado.votos++;
        contador++;
    } else {
        console.error("Error: Candidato no encontrado para votar.");
        return;
    }

    if (contador >= arrayDeAlumnos.length) {
        mostrarResultados();
        return;
    }

    const parrafo = document.getElementById("parrafo");
    parrafo.textContent = arrayDeAlumnos[contador].nombreCompleto + " es tu turno para votar";
}

function mostrarResultados() {
    document.getElementById("buttons-container").style.display = "none";
    document.getElementById("elegirCandidatos").textContent = "Resultados";

    const parrafoElemento = document.getElementById("parrafo");
    let contenidoResultado = "Votación finalizada. Resultados:<br><br>";

    const resultadosOrdenados = candidatos.sort((a, b) => b.votos - a.votos);

    resultadosOrdenados.forEach(candidato => {
        contenidoResultado += `${candidato.nombreCompleto}: ${candidato.votos} votos<br>`;
    });

    parrafoElemento.innerHTML = contenidoResultado;
}
```

Explicación:

Defino dos clases: Alumno (la base con id y nombreCompleto) y Candidato, que hereda las propiedades de Alumno y añade la funcionalidad específica para el conteo de votos (votos = 0). Utilizo Candidato extends Alumno para establecer esta relación de herencia. La función cargarArray() lee el fichero de alumnos con la API fetch y luego, mediante métodos de array (map y filter), creo instancias de la clase Alumno a partir de las líneas del fichero, almacenándolas en arrayDeAlumnos.

En seleccionCandidato(), obtengo el objeto Alumno base del array principal mediante su data-id y creo una nueva instancia de la clase Candidato a partir de ese objeto Alumno (patrón de delegación/adaptación), la cual añado al array candidatos. Este diseño permite separar la información base del alumno de la funcionalidad de votación (el conteo), mejorando la organización y la reusabilidad del código al trabajar con objetos en lugar de arrays simples en todas las fases del proceso.

BIBLIOGRAFÍA

MDN Web Docs (Mozilla Developer Network) Uso: Consulto la documentación esencial sobre Clases en JavaScript, incluyendo el uso de constructor y la palabra clave extends para implementar la herencia entre Candidato y Alumno. También reviso el manejo de promesas con fetch para la carga asíncrona de datos desde el archivo, fundamental en este diseño.
Enlaces:
https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Classes
https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch

Referencias de JavaScript Avanzado (Programación Orientada a Objetos) Uso: Verifico la sintaxis para crear nuevas instancias de objetos (new Alumno(), new Candidato()) y la forma de iterar sobre arrays de objetos usando forEach y sort() para presentar los resultados finales de la votación de forma ordenada.
Enlace:
https://developer.mozilla.org/es/docs/Web/JavaScript/Guide