Adrián Polo 17 de enero de 2025

Adrián Polo 17 de enero de 2025

UT7 PRÁCTICA 4

HTML

```html
<body>

    <h1>Consulta de Ingresos y Gastos</h1>

    <!-- Botón para realizar la consulta -->
    <button id="btnConsultar">Consultar</button>

    <!-- Aquí se mostrarán los checkboxes de Ingresos y Gastos -->
    <div id="checkboxes" style="display: none;">
      <label>
          <input type="checkbox" id="ingreso" value="Ingreso"> Ingreso
      </label>
      <label>
          <input type="checkbox" id="gasto" value="Gasto"> Gasto
      </label>

      <select id="conceptosSelect">
          <option value="">Seleccione un concepto</option>
      </select>
      <br><br>
    </div>

    <!-- Mostrar los resultados de la consulta -->
    <div id="resultado">
        <!-- Los resultados de la consulta aparecerán aquí -->
    </div>


</body>
```

Es igual que el ejercicio anterior pero con un select donde se mostrarán los conceptos por los cuales se filtrará.

Consulta de Ingresos y Gastos

Consultar

☐ Ingreso ☐ Gasto Seleccione un concepto

JS

```
class GastosIngresos {
  // Constructor para inicializar el objeto
  constructor(Id, IngresoGasto, Valor, Descripcion, Fecha, IdConcepto) {
      this._Id = Id || null;  // ID del gasto (puede ser null si es un
nuevo gasto)
      this._IngresoGasto = IngresoGasto || '';  // 'Ingreso' o 'Gasto'
      this._Valor = Valor || 0.00;  // Monto del gasto o ingreso
      this._Descripcion = Descripcion || '';  // Descripción del gasto
      this._Fecha = Fecha || '';  // Fecha en formato 'YYYY-MM-DD'
      this._IdConcepto = IdConcepto || 0;  // Id del concepto relacionado
  }

  // Getter y Setter para _Id
  get Id() {
      return this._Id;
  }

  set Id(value) {
      this._Id = value;
  }

  // Getter y Setter para _IngresoGasto
  get IngresoGasto() {
      return this._IngresoGasto;
  }

  set IngresoGasto(value) {
      this._IngresoGasto = value;
  }

  // Getter y Setter para _Valor
  get Valor() {
      return this._Valor;
  }

  set Valor(value) {
      if (value >= 0) {
          this._Valor = value;
```

```
        } else {
            console.error("El valor debe ser positivo.");
        }
    }

    // Getter y Setter para _Descripcion
    get Descripcion() {
        return this._Descripcion;
    }

    set Descripcion(value) {
        this._Descripcion = value;
    }

    // Getter y Setter para _Fecha
    get Fecha() {
        return this._Fecha;
    }

    set Fecha(value) {
        this._Fecha = value;
    }

    // Getter y Setter para _IdConcepto
    get IdConcepto() {
        return this._IdConcepto;
    }

    set IdConcepto(value) {
        if (Number.isInteger(value) && value > 0) {
            this._IdConcepto = value;
        } else {
            console.error("El Id del concepto debe ser un número entero
positivo.");
        }
    }
}
```

Tenemos la clase para gastos e ingresos, igual que el ejercicio anterior.

```
window.onload = function() {

  // Obtener el botón "Consultar"
  var btnConsultar = document.getElementById('btnConsultar');

  // Obtener los checkboxes de ingreso y gasto
  var ingresoCheckbox = document.getElementById('ingreso');
  var gastoCheckbox = document.getElementById('gasto');

  // Obtener el contenedor de los checkboxes
  var checkboxesDiv = document.getElementById('checkboxes');

  // Obtener el div de resultados
  var resultadoDiv = document.getElementById('resultado');

  const select = document.getElementById('conceptosSelect');

  // Event listener para el botón "Consultar"
  btnConsultar.addEventListener('click', function() {
      this.disabled = true;
      // Mostrar los checkboxes para seleccionar Ingresos o Gastos
      checkboxesDiv.style.display = 'block';
  });
```

hacemos las referencias a elementos del html

```
cargarConceptos();
// Event listener para los cambios en los checkboxes
ingresoCheckbox.addEventListener('change', actualizarTabla);
gastoCheckbox.addEventListener('change', actualizarTabla);

select.addEventListener('change', actualizarTabla);
//document.getElementById('conceptosSelect').addEventListener('change', cargarGastos);
// Función para obtener y actualizar los datos según los checkboxes seleccionados
function actualizarTabla() {

    // Crear un array con los tipos seleccionados
    var tiposSeleccionados = [];

    if (ingresoCheckbox.checked) {
        tiposSeleccionados.push('Ingreso');
    }
    if (gastoCheckbox.checked) {
        tiposSeleccionados.push('Gasto');
    }

    // Si no se selecciona ningún tipo, limpiar los resultados
    if (tiposSeleccionados.length === 0) {
        resultadoDiv.innerHTML = '';  // Limpiar los resultados
        return;
    }

    let datoSeleccionado = select.options[select.selectedIndex].textContent;

    // Realizar la solicitud AJAX para obtener los datos
    obtenerDatos(tiposSeleccionados, datoSeleccionado);
}
```

Cargamos los conceptos de la base de datos en el select para posteriormente seleccionarlos.

```
// Función para realizar la solicitud AJAX
function obtenerDatos(tiposSeleccionados, datoSeleccionado) {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'GastosObtenerTodos.php', true);
    xhr.setRequestHeader('Content-Type', 'application/json');

    xhr.onload = function () {
        if (xhr.status === 200 && this.status == 4) {
            var response = JSON.parse(xhr.responseText);
            let gastosIngresos = response.map(item => {
                return new GastosIngresos(
                    item.Id,
                    item.Ingreso_gasto,
                    item.Valor,
                    item.Descripcion,
                    item.Fecha,
                    item.Id_concepto
                );
            });

            // Limpiar los resultados anteriores
            resultadoDiv.innerHTML = '';

            // Crear la tabla
            var tabla = document.createElement('table');
            tabla.id = 'tablaResultados';

            // Crear el encabezado de la tabla
            var thead = document.createElement('thead');
            var trHead = document.createElement('tr');
            ['ID', 'Operación', 'Valor', 'Descripción', 'Fecha', 'Concepto'].forEach(texto => {
                var th = document.createElement('th');
                th.textContent = texto;
                trHead.appendChild(th);
            });
            thead.appendChild(trHead);
            tabla.appendChild(thead);
```

Igual que el ejercicio anterior, hacemos la petición AJAX y guardamos todos los datos de la base de datos en el objeto gastos ingresos, creado con la clase anterior.

```
// Crear el cuerpo de la tabla
var tbody = document.createElement('tbody');

// Crear un array de promesas para manejar asincronía
let promesas = gastosIngresos.flatMap(item =>
    tiposSeleccionados.map(tipo => {
        if (item.IngresoGasto === tipo) {
            return obtenerId(datoSeleccionado).then(conceptoId => {
                if (conceptoId === item.IdConcepto || conceptoId === null) {
                    var tr = document.createElement('tr');

                    [item.Id, item.IngresoGasto, item.Valor, item.Descripcion, item.Fecha, ite
                        var td = document.createElement('td');
                        td.textContent = valor;
                        tr.appendChild(td);
                    });

                    tbody.appendChild(tr);
                }
            });
        }
    })
);

// Esperar a que todas las promesas se resuelvan antes de agregar la tabla al `resultadoDiv`
Promise.all(promesas).then(() => {
    tabla.appendChild(tbody);
    resultadoDiv.appendChild(tabla);
});
```

Creamos la tabla y recorremos todos los datos devueltos con el flatmap, y el filtrado es igual que al otro ejercicio pero añadiendo si el concepto existe o no. Recorremos todas las promesas guardadas y las mostramos en la tabla.

```
// Función para obtener conceptos desde el servidor y mostrarlos en un select
function cargarConceptos() {
    fetch('ConceptosObtenerTodos.php')
        .then(response => response.json())
        .then(data => {
            const select = document.getElementById('conceptosSelect');
            // Limpiar el select antes de agregar nuevas opciones
            select.innerHTML = '<option value="">Seleccione un concepto</option>';
            data.forEach(concepto => {
                const option = document.createElement('option');
                option.value = concepto.id;
                option.textContent = concepto.nombre; // Mejor usar textContent en lugar de .text
                select.appendChild(option);
            });
        })
        .catch(error => console.error('Error cargando conceptos:', error));
}
```

Y asi cargamos los conceptos en el select.

```
async function obtenerId(nombreConcepto) {
  try {
      const response = await fetch('ConceptosObtenerTodos.php');
      const data = await response.json();

      // Buscar el concepto por nombre
      const concepto = data.find(concepto => concepto.nombre === nombreConcepto);

      return concepto ? concepto.id : null; // Retorna el ID si se encuentra, si no, null
  } catch (error) {
      console.error('Error cargando conceptos:', error);
      return null; // Retorna null en caso de error
  }
```

Con esto obtenemos el id del concepto, para poder filtrarlo.