Código para Manejar terjetas con Fetch y JSON para examen

Te separo el código en partes bien explicadas y comentadas:

0. Datos de ejemplo para la tarjeta de ejemplo de mas abajo:


```
[
  {
    "id": 1,
    "emoji": "🚀",
    "titulo": "Proyecto Alpha",
    "descripcion": "Desarrollo de aplicación web para gestión de
inventario",
    "fecha": "2025-05-15",
    "fechaFormateada": "15-05-2025",
    "valoracion": 4.5,
    "categoria": "Desarrollo",
    "detalles": {
      "Categoría": "Desarrollo",
      "Autor": "Juan Pérez",
      "Valoración": "4.5/5"
    }
  },
  {
    "id": 2,
    "emoji": "🌟",
    "titulo": "Evento Anual",
    "descripcion": "Conferencia sobre tecnologías emergentes",
    "fecha": "2025-05-10",
    "fechaFormateada": "10-05-2025",
    "valoracion": 4.8,
    "categoria": "Eventos",
    "detalles": {
      "Categoría": "Eventos",
      "Autor": "María López",
      "Valoración": "4.8/5"
    }
  },
  {
    "id": 3,
    "emoji": "📊",
    "titulo": "Análisis de Datos",
    "descripcion": "Estudio estadístico del rendimiento trimestral",
    "fecha": "2025-05-03",
    "fechaFormateada": "03-05-2025",
    "valoracion": 3.7,
    "categoria": "Análisis",
    "detalles": {
```

```
      "Categoría": "Análisis",
      "Autor": "Pedro Gómez",
      "Valoración": "3.7/5"
    }
  },
  {
    "id": 4,
    "emoji": "🔍",
    "titulo": "Investigación de Mercado",
    "descripcion": "Estudio sobre tendencias de consumo",
    "fecha": "2025-04-28",
    "fechaFormateada": "28-04-2025",
    "valoracion": 4.2,
    "categoria": "Investigación",
    "detalles": {
      "Categoría": "Investigación",
      "Autor": "Ana Martínez",
      "Valoración": "4.2/5"
    }
  }
]
```

1. Función para Cargar los Datos JSON

```
/**
 * Carga los datos de Pokémon desde un archivo JSON local
 * @returns {Promise<Array>} Array con los datos de Pokémon o array vacío
si hay error
 */
async function loadPokemons() {
  try {
    // Hacemos la petición fetch (GET por defecto)
    const response = await fetch("http://localhost:3000/pokedex.json");

    // Verificamos si la respuesta es correcta (status 200-299)
    if (!response.ok) {
      throw new Error(`Error HTTP: ${response.status}`);
    }

    // Convertimos la respuesta a JSON
    const data = await response.json();
    return data;
  } catch (error) {
    // Manejo de errores (fallo en fetch, JSON mal formado, etc.)
    console.error("Error cargando Pokémon:", error);
    return []; // Devolvemos array vacío para que el programa no se rompa
  }
}
```

Otra opción:

```
// Función para cargar datos desde un JSON externo
function cargarDatosJSON(url) {
  fetch(url)
    .then((response) => response.json())
    .then((data) => {
      datosOriginales = data;
      datosFiltrados = [...data];
      cargarCategorias(data);
      cargarTarjetas(data);
    })
    .catch((error) => {
      console.error("Error al cargar los datos:", error);
    });
}
// Inicializar la aplicación cuando el DOM esté listo
document.addEventListener("DOMContentLoaded", () => {
  // Inicializar con los datos de ejemplo
  datosOriginales = ejemploJSON;
  datosFiltrados = [...ejemploJSON];

  // Cargar categorías para el filtro
  cargarCategorias(ejemploJSON);

  // Cargar las tarjetas iniciales
  cargarTarjetas(ejemploJSON);

  // Configurar eventos para los filtros
  inicializarEventos();

  // Si quieres cargar datos desde un JSON externo en lugar de usar el
ejemplo:
  // cargarDatosJSON('tu-archivo.json');
});
```

2. Función para Crear Tarjetas HTML

```
// Función para generar una tarjeta a partir de un objeto de datos
function crearTarjeta(datos) {
  const tarjeta = document.createElement("div");
  tarjeta.className = "card";

  // Construir detalles HTML
  let detallesHTML = "";
  for (const [clave, valor] of Object.entries(datos.detalles || {})) {
    detallesHTML += `
                    <div class="card-detail-item">
                        <span>${clave}:</span>
                        <span>${valor}</span>
                    </div>
                `;
```

```
  }

  // Construir estructura HTML de la tarjeta
  tarjeta.innerHTML = `
                <div class="emoji-container">${datos.emoji || "📄"}</div>
                <div class="card-title">${datos.titulo}</div>
                <div class="card-description">${datos.descripcion}</div>

                <div class="card-details">
                    ${detallesHTML}
                </div>

                <div class="card-footer">
                    <span>${datos.fecha}</span>
                    <button class="card-button" data-id="${
                      datos.id
                    }">Ver</button>
                </div>
            `;

  // Añadir evento al botón
  tarjeta.querySelector(".card-button").addEventListener("click", function
() {
    alert(`Has seleccionado la tarjeta con ID: ${datos.id}`);
    // Aquí puedes añadir tu lógica para manejar el clic
  });

  return tarjeta;
}
```

HTML Contenedor dinamico

```
<div class="cards-container" id="cards-container">
  <!-- Aquí se generarán dinámicamente las tarjetas -->
</div>
```

CSS de la tarjeta

```
/* CSS minimalista */
body {
  font-family: sans-serif;
  margin: 20px;
}

.cards-container {
  display: flex;
  flex-wrap: wrap;
  gap: 15px;
}
```

```css
.card {
  border: 1px solid #ddd;
  border-radius: 5px;
  width: 250px;
  padding: 15px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.emoji-container {
  font-size: 30px;
  text-align: center;
  margin-bottom: 10px;
}

.card-title {
  font-weight: bold;
  margin-bottom: 5px;
}

.card-description {
  color: #666;
  font-size: 14px;
  margin-bottom: 10px;
}

.card-details {
  border-top: 1px solid #eee;
  padding-top: 10px;
  font-size: 13px;
}

.card-detail-item {
  display: flex;
  justify-content: space-between;
  margin-bottom: 3px;
}

.card-footer {
  display: flex;
  justify-content: space-between;
  margin-top: 10px;
  align-items: center;
}

.card-button {
  background-color: #0078d7;
  color: white;
  border: none;
  border-radius: 3px;
  padding: 5px 10px;
  cursor: pointer;
}
```

3. Función Principal para Mostrar tarjetas

```
// Función para cargar las tarjetas al contenedor
function cargarTarjetas(datos) {
  const contenedor = document.getElementById("cards-container");
  contenedor.innerHTML = ""; // Limpiar el contenedor

  if (datos.length === 0) {
    const noResults = document.createElement("div");
    noResults.className = "no-results";
    noResults.textContent =
      "No se encontraron resultados con los filtros actuales";
    contenedor.appendChild(noResults);
    return;
  }

  datos.forEach((item) => {
    const tarjeta = crearTarjeta(item);
    contenedor.appendChild(tarjeta);
  });
}
```

4. Funciones de Filtrado

CSS de los elementos de filtrado

```
.controls {
  background: #f5f5f5;
  padding: 15px;
  margin-bottom: 20px;
  border-radius: 5px;
  border: 1px solid #ddd;
}

.filter-group {
  margin-bottom: 15px;
}

.filter-group label {
  display: block;
  margin-bottom: 5px;
  font-weight: bold;
}

.slider-container {
  display: flex;
  align-items: center;
}
```

```
.slider-container input {
  flex-grow: 1;
  margin-right: 10px;
}

.slider-value {
  min-width: 30px;
}

input[type="text"],
input[type="number"] {
  padding: 5px;
  width: 100%;
  box-sizing: border-box;
}

input[type="range"] {
  width: 100%;
}
.no-results {
  width: 100%;
  padding: 20px;
  text-align: center;
  color: #666;
  font-style: italic;
}
```

HTML de los elementos de filtrado

```
<!-- Controles de filtros -->
<div class="controls">
  <div class="filter-group">
    <label for="search-filter">Buscar por texto:</label>
    <input
      type="text"
      id="search-filter"
      placeholder="Escribe para filtrar..."
    />
  </div>

  <div class="filter-group">
    <label for="rating-filter">Valoración mínima:</label>
    <div class="slider-container">
      <input
        type="range"
        id="rating-filter"
        min="0"
        max="5"
        step="0.1"
        value="0"
      />
```

```
      <span class="slider-value" id="rating-value">0</span>
    </div>
  </div>

  <div class="filter-group">
    <label for="date-filter">Fecha desde:</label>
    <input type="date" id="date-filter" />
  </div>

  <div class="filter-group">
    <label for="category-filter">Categoría:</label>
    <select id="category-filter">
      <option value="">Todas</option>
      <!-- Se llenará dinámicamente -->
    </select>
  </div>
</div>
```

Función de filtrado

```
let datosOriginales = [...ejemploJSON];
let datosFiltrados = [...ejemploJSON];

// Función principal para filtrar datos según criterios múltiples
function filtrarDatos(datos, filtros) {
  return datos.filter((item) => {
    // Filtro de texto (busca en título y descripción)
    if (
      filtros.texto &&
      !item.titulo.toLowerCase().includes(filtros.texto.toLowerCase()) &&

!item.descripcion.toLowerCase().includes(filtros.texto.toLowerCase())
    ) {
      return false;
    }

    // Filtro de valoración mínima
    if (
      filtros.valoracionMinima &&
      item.valoracion < filtros.valoracionMinima
    ) {
      return false;
    }

    // Filtro de fecha mínima
    if (
      filtros.fechaDesde &&
      new Date(item.fecha) < new Date(filtros.fechaDesde)
    ) {
      return false;
    }
```

```
    // Filtro de categoría
    if (filtros.categoria && item.categoria !== filtros.categoria) {
      return false;
    }

    return true;
  });
}

// Función para obtener los filtros actuales
function obtenerFiltrosActuales() {
  return {
    texto: document.getElementById("search-filter").value,
    valoracionMinima: parseFloat(
      document.getElementById("rating-filter").value
    ),
    fechaDesde: document.getElementById("date-filter").value,
    categoria: document.getElementById("category-filter").value,
  };
}

// Función para aplicar los filtros y actualizar la vista
function aplicarFiltros() {
  const filtros = obtenerFiltrosActuales();
  datosFiltrados = filtrarDatos(datosOriginales, filtros);
  cargarTarjetas(datosFiltrados);
}
// Función para cargar las opciones de categoría en el select
function cargarCategorias(datos) {
  const categorias = [...new Set(datos.map((item) => item.categoria))];
  const selectCategorias = document.getElementById("category-filter");

  categorias.forEach((categoria) => {
    const option = document.createElement("option");
    option.value = categoria;
    option.textContent = categoria;
    selectCategorias.appendChild(option);
  });
}
```

5. Inicialización y Eventos

```
// Función para inicializar los controladores de eventos
function inicializarEventos() {
  // Evento para búsqueda por texto (con debounce)
  let timeoutId;
  document
    .getElementById("search-filter")
    .addEventListener("input", function () {
      clearTimeout(timeoutId);
```

```
        timeoutId = setTimeout(aplicarFiltros, 300); // Debounce de 300ms
      });

    // Evento para slider de valoración
    document
      .getElementById("rating-filter")
      .addEventListener("input", function (e) {
        document.getElementById("rating-value").textContent =
  e.target.value;
        aplicarFiltros();
      });

    // Evento para filtro de fecha
    document
      .getElementById("date-filter")
      .addEventListener("change", aplicarFiltros);

    // Evento para filtro de categoría
    document
      .getElementById("category-filter")
      .addEventListener("change", aplicarFiltros);
  }
```