Incluir total compras

1. Transforma los precios a números al momento de calcular el total.
2. Agrega el total en la sección del carrito.

Incluir botón eliminar

// Función para eliminar un producto del carrito basado en su índice
// opción 1 Elimina basado en el índice del producto
// const handleRemoveFromCart = (product) => {
// setCart(cart.filter((item) => item.id !== product.id));
// };
<button className="btn btn-danger" onClick={() => handleRemoveFromCart(item)}>
Eliminar

// opción 2 eliminar un solo elemento basado en su índice en la lista del carrito.
const handleRemoveFromCart = (indexToRemove) => {
setCart(cart.filter((_, index) => index !== indexToRemove));
// Crea un nuevo array sin el producto que tiene el índice igual a `indexToRemove`
};

```
<button
       className="btn btn-danger"
       onClick={() => handleRemoveFromCart(index)}
     >
       Eliminar <FaTrashAlt />
     </button>
```

AGREGAR Y AGRUPAR POR PRODUCTO

Contexto
Esta función se utiliza para agregar un producto (en este caso, un Pokémon) al carrito de compras. Si el producto ya existe en el carrito, en lugar de agregarlo nuevamente, se incrementa la cantidad de ese producto. Si no existe, se agrega como un nuevo ítem con una cantidad inicial de 1.

Desglose del Código

const handleAddToCart = (pokemon) => {
setCart((prevCart) => {
// Busca si el producto ya existe en el carrito
const existingProduct = prevCart.find((item) => item.id === pokemon.id);

```
if (existingProduct) {
  // Si el producto ya está en el carrito, incrementa la cantidad
  return prevCart.map((item) =>
    item.id === pokemon.id
```

```
        ? { ...item, quantity: item.quantity + 1 } // Incrementa la cantidad en 1
        : item // No modifica otros productos en el carrito
    );
  } else {
    // Si el producto no está en el carrito, agrégalo con cantidad 1
    return [...prevCart, { ...pokemon, quantity: 1 }];
  }
```

});
};
Explicación Paso a Paso
Entrada de la función:

La función handleAddToCart toma un parámetro pokemon, que representa el producto (o Pokémon) que se quiere agregar al carrito.
Actualización del carrito con setCart:

setCart es la función que se usa para actualizar el estado del carrito (cart). Llama a esta función pasando una función de actualización que toma como argumento el estado previo del carrito (prevCart).
Buscar si el producto ya existe en el carrito:

prevCart.find((item) => item.id === pokemon.id) busca en el carrito previo (prevCart) un producto que tenga el mismo id que el producto que se quiere agregar. Si lo encuentra, lo almacena en la variable existingProduct.
Si existingProduct es undefined, significa que el producto no está en el carrito.
Producto existente en el carrito:

Si el producto ya existe (if (existingProduct)):
Se utiliza prevCart.map para crear un nuevo arreglo de productos basado en el carrito anterior.
Dentro de map, se verifica si el id del producto actual (item.id) es igual al id del producto que se está agregando (pokemon.id):
Si es así, se retorna un nuevo objeto con todos los datos del producto (...item) pero con la quantity incrementada en 1 (item.quantity + 1).
Si no es el producto que estamos agregando, se retorna el producto tal como está en el carrito, sin modificarlo.
Producto nuevo en el carrito:

Si el producto no existe en el carrito (else):
Se retorna un nuevo arreglo que es una copia del carrito previo ([...prevCart]), pero con el nuevo producto agregado al final ({ ...pokemon, quantity: 1 }).
Al producto se le agrega una nueva propiedad quantity con un valor inicial de 1.
Resultado:

Al final, la función retorna un nuevo estado para el carrito, ya sea con la cantidad de un producto incrementada o con un nuevo producto agregado.
Ejemplo
Caso 1: Agregas un "Pikachu" al carrito por primera vez.

existingProduct es undefined, entonces el código entra en el else y agrega "Pikachu" al carrito con quantity: 1.
Caso 2: Agregas "Pikachu" de nuevo.

existingProduct ya no es undefined, entonces el código entra en el if y usa map para incrementar la cantidad de "Pikachu" en el carrito.
Este enfoque garantiza que cada producto en el carrito sea único y que el número de unidades de cada producto se maneje de manera adecuada.

```
// const handleAddToCart = (pokemon) => {
//     setCart([...cart, pokemon]);
    // console.log(pokemon)
// };
```

```
    <!-- <ul>
      {cart.map((item, index) => (
        <li
          key={index}>{item.name} - {item.price}
          <button className="btn btn-danger" onClick={() =>
handleRemoveFromCart(index)}>
            Eliminar <FaTrashAlt />
          </button>
        </li>
      ))}
    </ul> -->
```

Detalles sobre parseFloat
Sintaxis:

parseFloat(string);
Parámetro:

string: Este es el valor que quieres convertir a un número de punto flotante. Puede ser una cadena de texto que contenga un número, y parseFloat extraerá el valor numérico de esa cadena.
Retorno:

Devuelve el número de punto flotante contenido en la cadena. Si la primera parte del texto no se puede convertir a un número, devuelve NaN (Not a Number).

usando parseFloat para convertir el precio de un producto, que está en formato de cadena con un símbolo de dólar ("$10.00"), en un número de punto flotante para poder realizar operaciones matemáticas con él, como multiplicarlo por la cantidad:

javascript
Copiar código
parseFloat(item.price.replace("$", ""))

item.price.replace("$", ""): Primero se elimina el símbolo de dólar de la cadena para que quede solo el número ("10.00").
parseFloat: Luego, parseFloat convierte esa cadena ("10.00") en un número (10.00) que se puede usar para cálculos.
Este número se puede multiplicar por la cantidad de productos en el carrito para calcular el precio total para ese producto específico.