Guardar los ítems del carrito en el sessionStorage

En esta sección, se abordará la implementación del manejo de sesión en el navegador. Además de gestionar el estado de los productos en React, se utilizará el **sessionStorage** para almacenar la información del carrito en la sesión del navegador. Esto permitirá que los productos agregados permanezcan incluso después de recargar la página.

Pasos 54 al 58

Sigue estos pasos:

Paso 54: Implementar sessionStorage para el almacenamiento de sesión

Según la documentación de Mozilla Developer. **sessionStorage** es una propiedad nativa del navegador en JavaScript que permite acceder a un objeto de almacenamiento asociado a la sesión actual del usuario.

A diferencia de **localStorage**, **sessionStorage** tiene una vida útil limitada a la duración de la sesión del navegador. Esto significa que los datos almacenados en **sessionStorage** se eliminan automáticamente cuando se cierra la ventana del navegador. Si se requiere que la información persista incluso después de cerrar el navegador, se debe utilizar **localStorage**. (<https://developer.mozilla.org/es/docs/Web/API/Window/sessionStorage>)

La sintaxis para trabajar con sessionStorage es sencilla:

// Almacena la información en sessionStorage

sessionStorage.setItem("key", "value");

// Obtiene la información almacenada desde sessionStorage

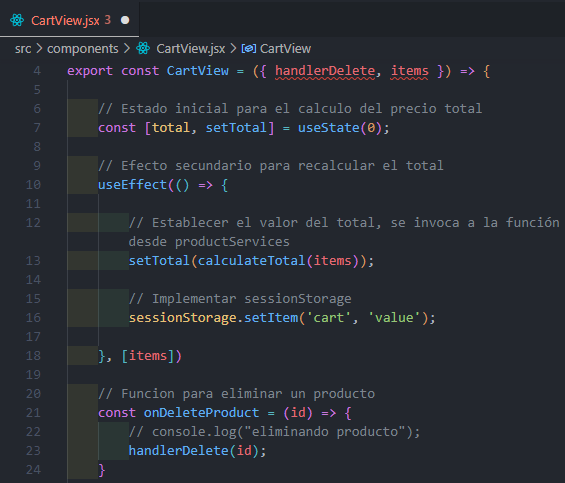
var data = sessionStorage.getItem("key");

Para almacenar datos, se utiliza el método **setItem**, que toma dos argumentos: una clave (**key**) y un valor (**value**).

Para recuperar la información almacenada, se utiliza el método **getItem**, que recibe como argumento la clave correspondiente.

En el contexto de esta implementación; dentro del componente **CartView**, en el efecto secundario de **useEffect** que recalcula el total cuando cambian los ítems, **sessionStorage** se utilizará para mantener el estado del carrito de compras del usuario.

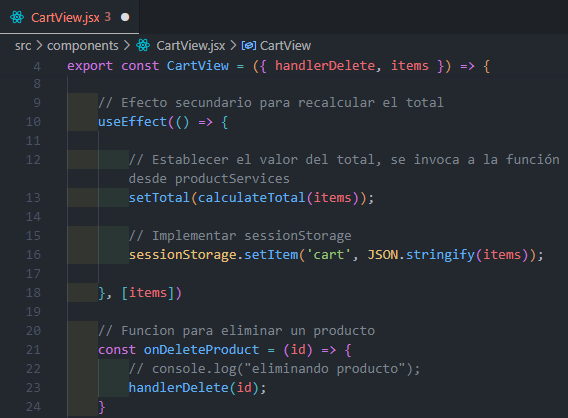
Cada vez que se actualicen los elementos del carrito con el método **setItem** (no es una función de un estado de React), se almacenará la información actualizada en **sessionStorage**, lo que garantiza que los productos agregados al carrito permanezcan disponibles incluso después de recargar la página.



Paso 55: Convertir el valor de un objeto recibido a un string

Almacenar objetos en **sessionStorage** requiere que se conviertan en cadenas de texto, ya que **sessionStorage** solo puede almacenar valores de tipo String. En JavaScript, se puede lograr esto utilizando el método **JSON.stringify** que proporciona el objeto global JSON. Este método toma un objeto como argumento y lo convierte en una cadena de caracteres JSON.

Siguiendo el paso anterior, se puede implementar la conversión del objeto de **items** a una cadena de texto utilizando **JSON.stringify**. Luego para almacenar esta cadena en **sessionStorage** se utiliza **sessionStorage.setItem**.



Después de esta implementación, los ítems del carrito se guardarán en **sessionStorage** cada vez que cambien. Ahora se necesita asegurar de que, al iniciar la aplicación, los ítems se recuperen correctamente desde **sessionStorage** y se asignen al estado inicial de **items**.

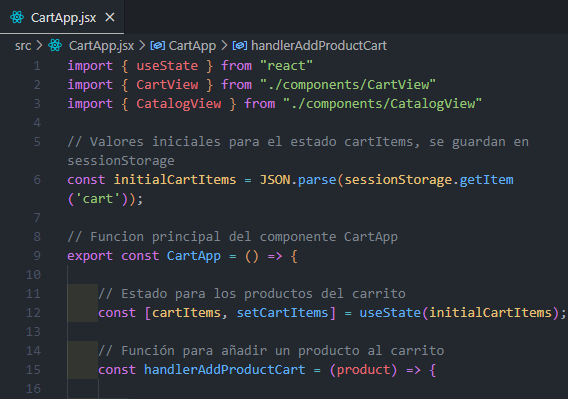
En el componente principal **CartApp**, se puede inicializar el estado de los ítems del carrito utilizando **sessionStorage.getItem** y asignando el valor recuperado a la constante **initialCartItems**. Luego, se puede utilizar esta constante como el estado inicial de los ítems del carrito.



Paso 56: Convertir el valor de un string recibido a un objeto

Después de almacenar los ítems del carrito como una cadena de texto en **sessionStorage**, se necesita recuperarlos y convertirlos de nuevo a un objeto cuando se inicializa la aplicación.

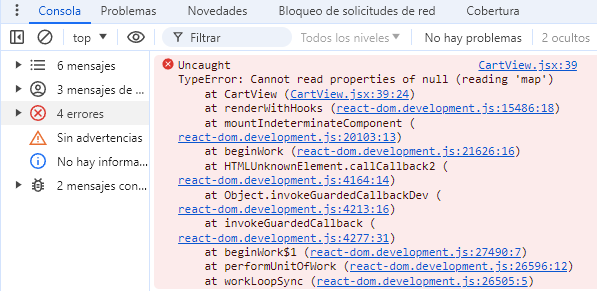
Para lograr esto, se puede utilizar **JSON.parse**, un método que convierte una cadena de caracteres JSON en un objeto JavaScript. Dentro del componente principal **CartApp**, podemos utilizar **JSON.parse** para convertir el valor recuperado de **sessionStorage** de nuevo a un objeto y asignarlo al estado inicial de los ítems del carrito.



Es importante recordar que **JSON.parse** convierte una cadena de caracteres JSON en un objeto, mientras que **JSON.stringify** convierte un objeto JavaScript en una cadena de caracteres JSON. Esto nos permite almacenar y recuperar datos complejos de manera efectiva utilizando **sessionStorage**.

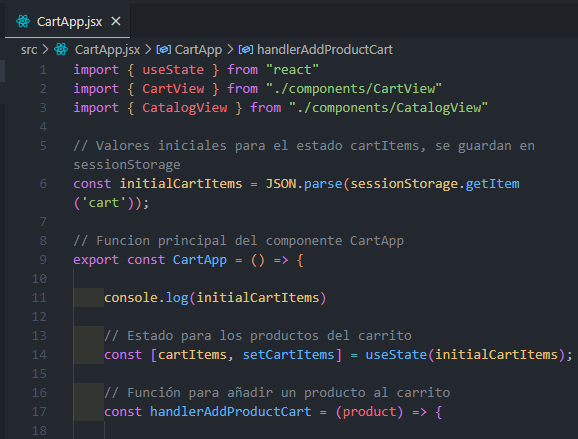
Al realizar una prueba en el navegador, se puede observar un error en la consola que indica que **initialCartItems** está vacío, lo cual provoca un problema al intentar utilizar el método **map** en los ítems del carrito. Este error se produce porque no se han agregado ítems a la sesión en **sessionStorage** durante la ejecución de la aplicación.

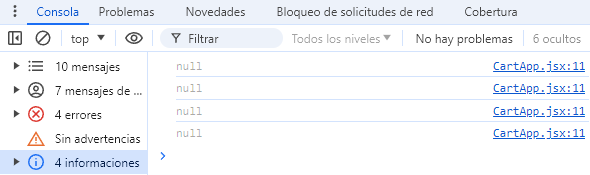
La razón detrás de esto es que, hasta ahora, los ítems del carrito se han almacenado en **sessionStorage** solo cuando cambia el estado del carrito. Por lo tanto, si no se han agregado ítems al carrito durante la sesión actual, **initialCartItems** permanecerá vacío, lo que provocará el error mencionado.



Paso 57: Establecer el valor inicial de initialCartItems mediante una condición

Comienza imprimiendo el valor de **initialCartItems** para comprender su estado inicial en la consola del navegador. Al hacerlo, se observa que inicialmente devuelve **null**, lo cual es obvio dado que aún no se ha almacenado ningún producto en la sesión.





Se procede a eliminar el **console.log** definido previamente.

Para evitar posibles errores al intentar acceder a **initialCartItems** cuando es **null**, se utiliza el operador lógico **or** (**||**) para establecer un valor predeterminado en caso de que **initialCartItems** sea **null**. Si **initialCartItems** es **null**, se asigna un arreglo vacío como valor predeterminado.

Esta estrategia va a permitir inicializar correctamente **initialCartItems**, garantizando que siempre tenga un valor válido, ya sea el contenido almacenado en **sessionStorage** o un arreglo vacío si no hay datos disponibles. Como resultado, se evita posibles errores y se puede trabajar con el estado del carrito de compras de manera segura y consistente en nuestra aplicación.

