Form, desacoplar eventos en funciones

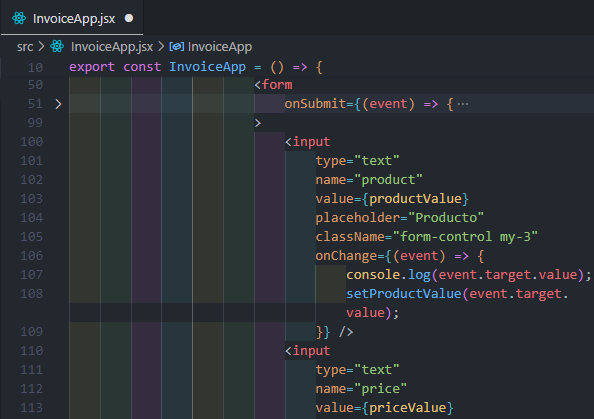
En esta sección, se procederá a limpiar el código HTML, de tal manera que las funciones definidas directamente en los eventos sean trasladadas. Para aquello, se trasladará el código de las funciones a la parte JavaScript del componente, con el fin de mejorar la organización y legibilidad del código.

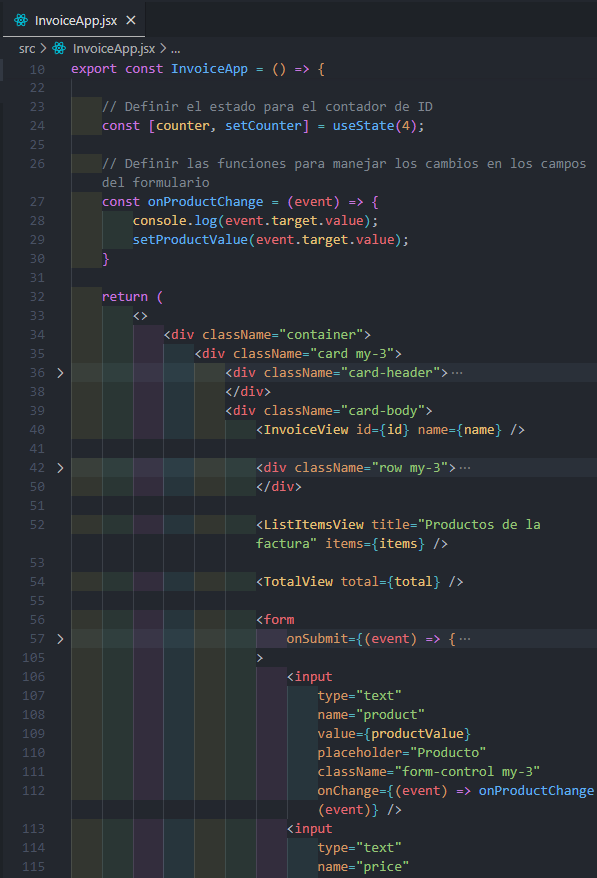
Pasos 75 al 78

Sigue estos pasos:

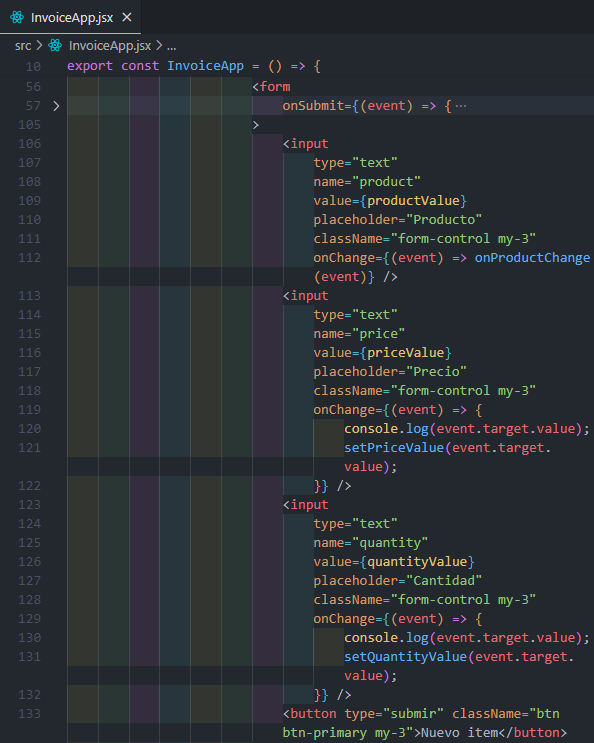
Paso 75: Desacoplar el evento en los campos del formulario

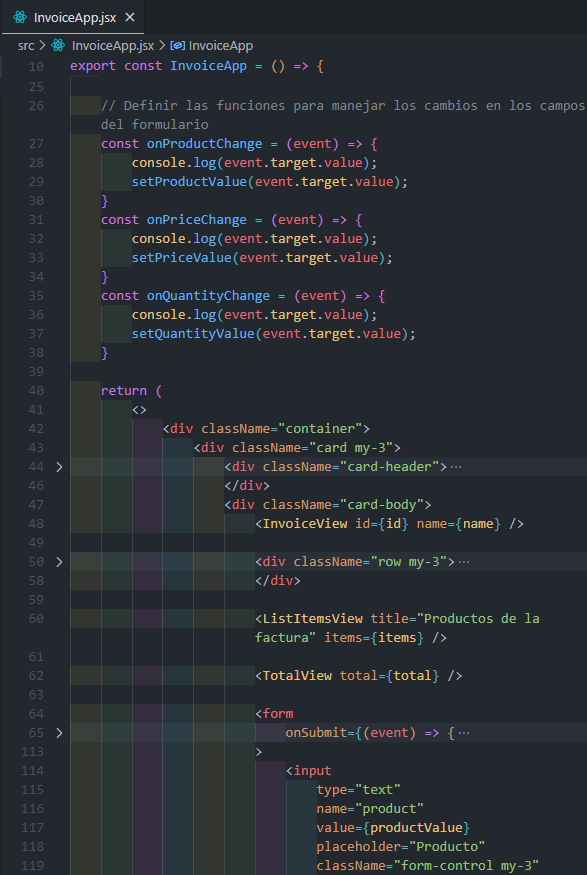
Primero se va a empezar con el campo para introducir el nombre del producto (**product**). Procede a trasladar el código del evento **onChange** a una función separada dentro del componente, en este caso hacia la función de flecha **onProductChange** definida antes del **return** con el parámetro de tipo objeto **event**.





Realiza el mismo procedimiento con los 2 campos restantes del formulario (**price** y **quantity**). Se corta el código definido en el evento del campo del formulario y se pega en el cuerpo de la función definida para el campo correspondiente.

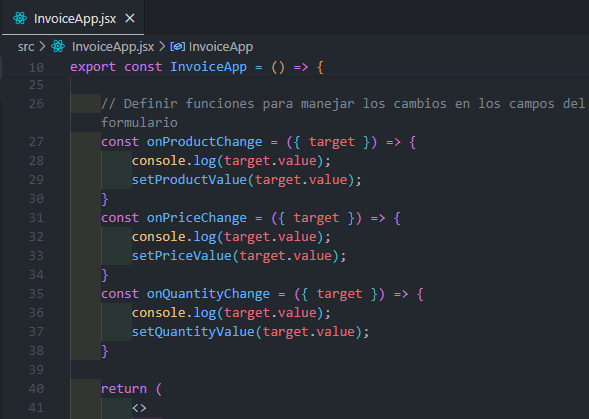


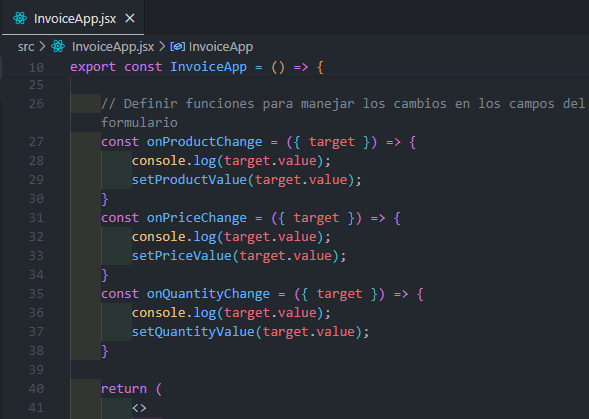




Paso 76: Simplificar el código para mejorar la eficacia

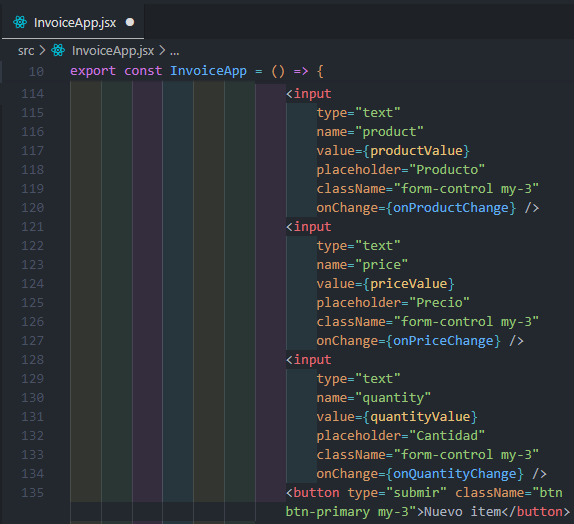
Para mejorar el manejo de eventos, se procede a desestructurar el objeto **event** para usar directamente la propiedad **target**. Se simplifica el código y se hace más clara la intención de las funciones.





**Nota:** Recuerda que puedes utilizar multicursor en Visual Studio Code para escribir el mismo código en varias líneas de manera simultánea, solamente presiona la tecla **ALT** y luego marca con un clic los "puntos" donde se va a añadir el mismo código.

Luego se puede eliminar la necesidad de pasar el parámetro **event** al llamar a la función definida, dentro del evento **onChange** de los campos del formulario, esto se logra al pasar la función directamente, por ejemplo en el campo del precio del producto (**price**): **onChange={onPriceChange}**.

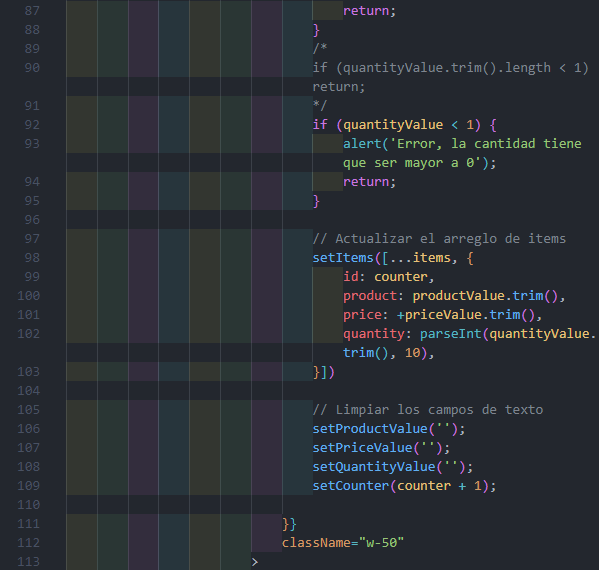


Es importante tener en cuenta que si se pasa el parámetro **event**, al usar **onChange={(event) => onPriceChange(event)}**, se crea una nueva función anónima cada vez que el componente se renderiza. Esto puede tener un impacto en el rendimiento en aplicaciones más grandes, ya que se generan más objetos de función innecesarios.

Paso 77: Desacoplar el evento onSubmit del formulario

Desacopla el evento **onSubmit** del elemento **<form>**. Para aquello procede a cortar todo el contenido definido en la función de flecha del evento **onSubmit**:



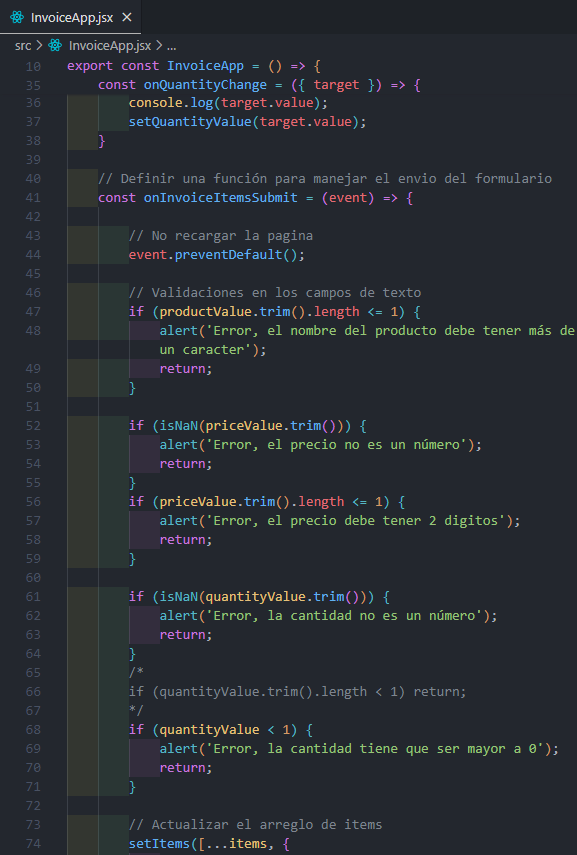


Luego crea una nueva función en una constante llamada **onInvoiceItemsSubmit** que toma el argumento **event**. En el cuerpo de la función se pega el código previamente extraído dentro de esta nueva función.

Es importante mencionar que se mantiene el uso del objeto **event** sin desestructurar, ya que es necesario para llamar a **preventDefault**.

Finalmente se reemplaza la función de flecha en el evento **onSubmit** del formulario (elemento **<form>**) por la referencia directa a **onInvoiceItemsSubmit**. Solamente se utiliza el nombre de la función como referencia.

**Nota:** Ten en cuenta que la función encargada de enviar el formulario tiene la sintaxis de empezar con el nombre "on" y luego unas palabras que tenga relación con lo que se espera del formulario.





Con estos cambios, se ha logrado desacoplar correctamente los eventos de los elementos y simplificado el código, mejorando su legibilidad y mantenibilidad.

Paso 78: Realizar una prueba con el formulario

En conclusión, todo el código está mucho más organizado. Ahora, todo el código JavaScript se encuentra antes del **return** dentro del componente de la función, mientras que todo el código HTML está luego del **return**. En el HTML, los eventos del formulario, como el **onChange**, hacen llamadas a las funciones. Estas funciones, a su vez, llaman a otras funciones y pasan el argumento **event** en las cuatro funciones (**onInvoiceItemsSubmit**, **onProductChange**, **onPriceChange**, **onQuantityChange**).

Todo funciona tal cual como antes: no hay errores en la consola y se crea el producto correctamente (sin haber cometido los errores de validación mencionados previamente).

