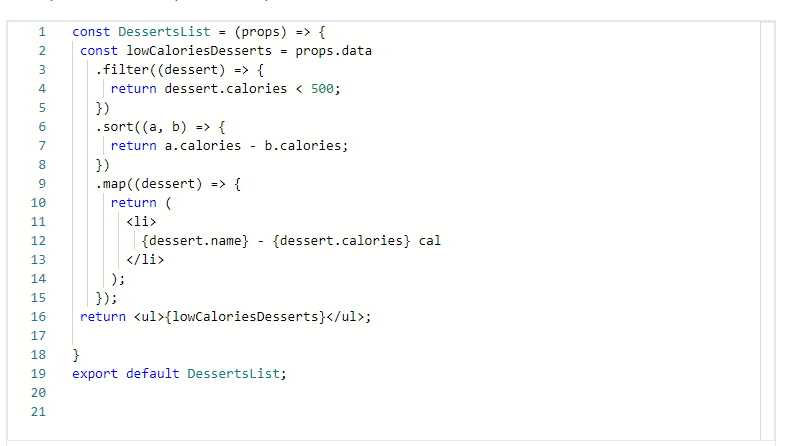
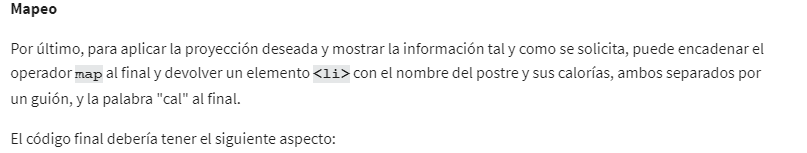
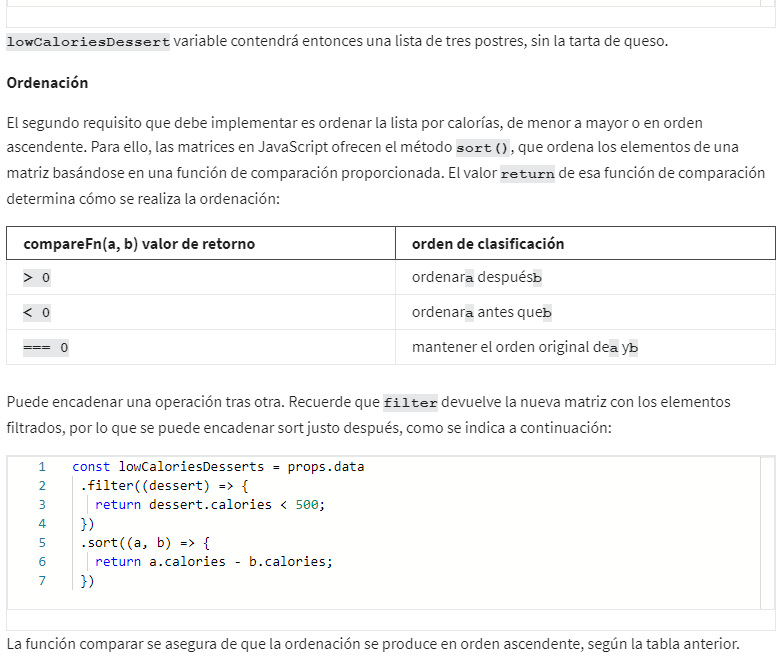
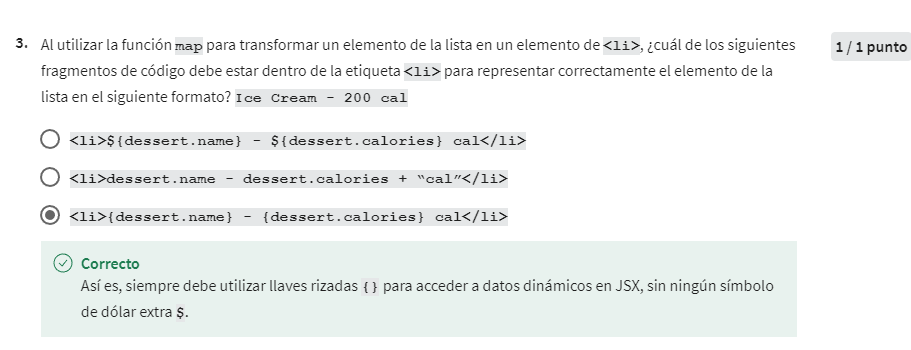
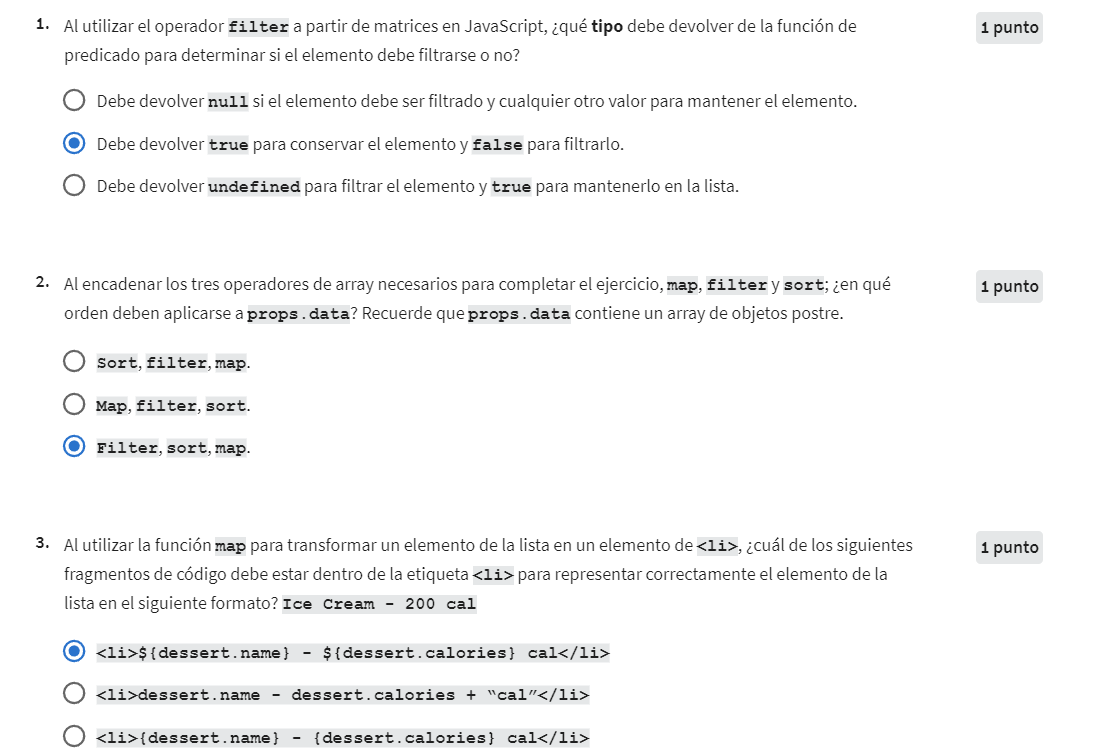
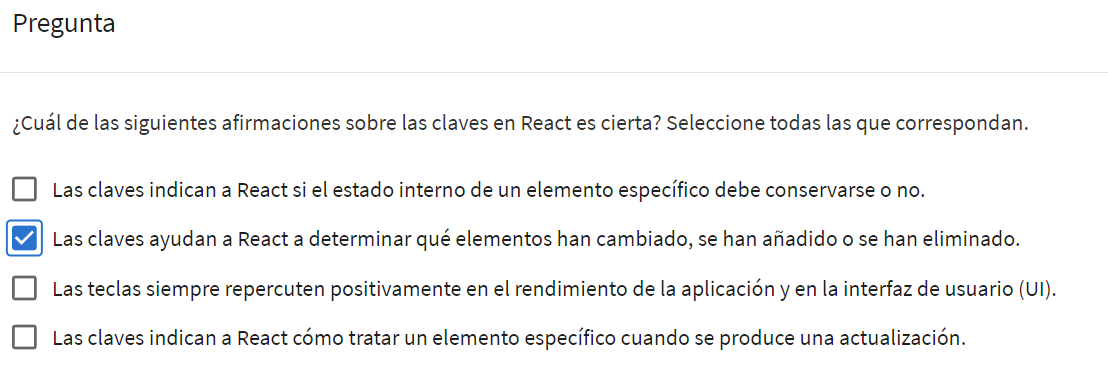
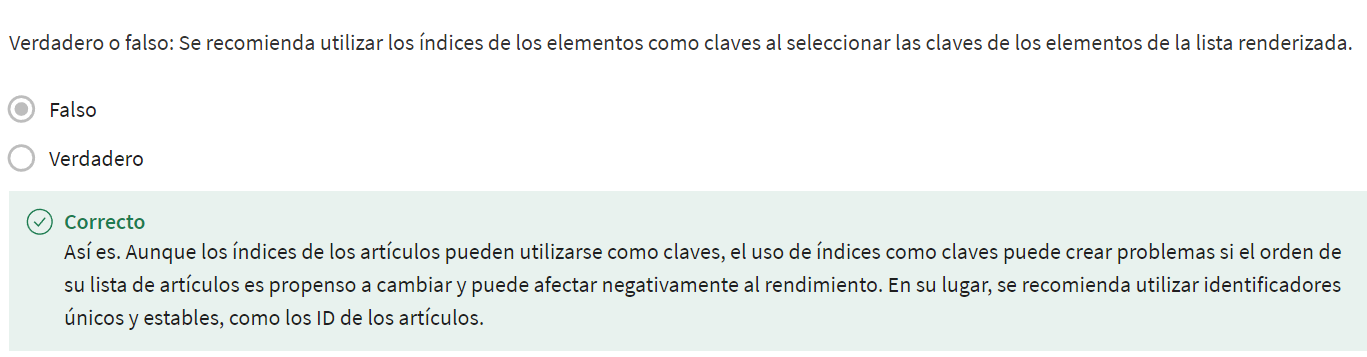
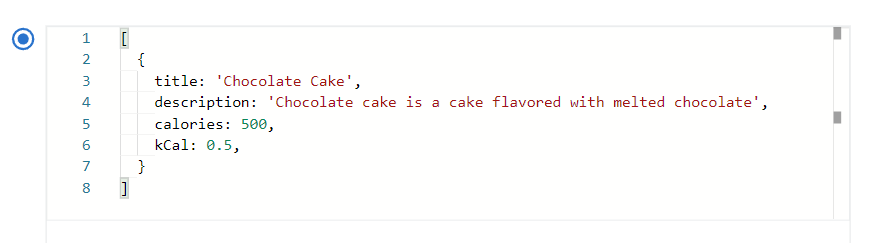


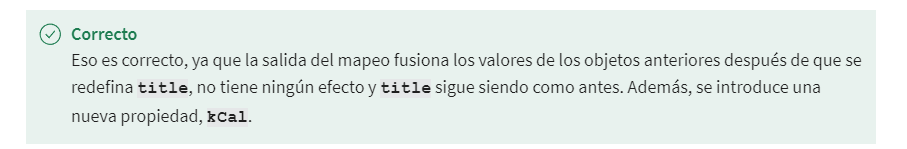
**Crear un componente básico de Lista**

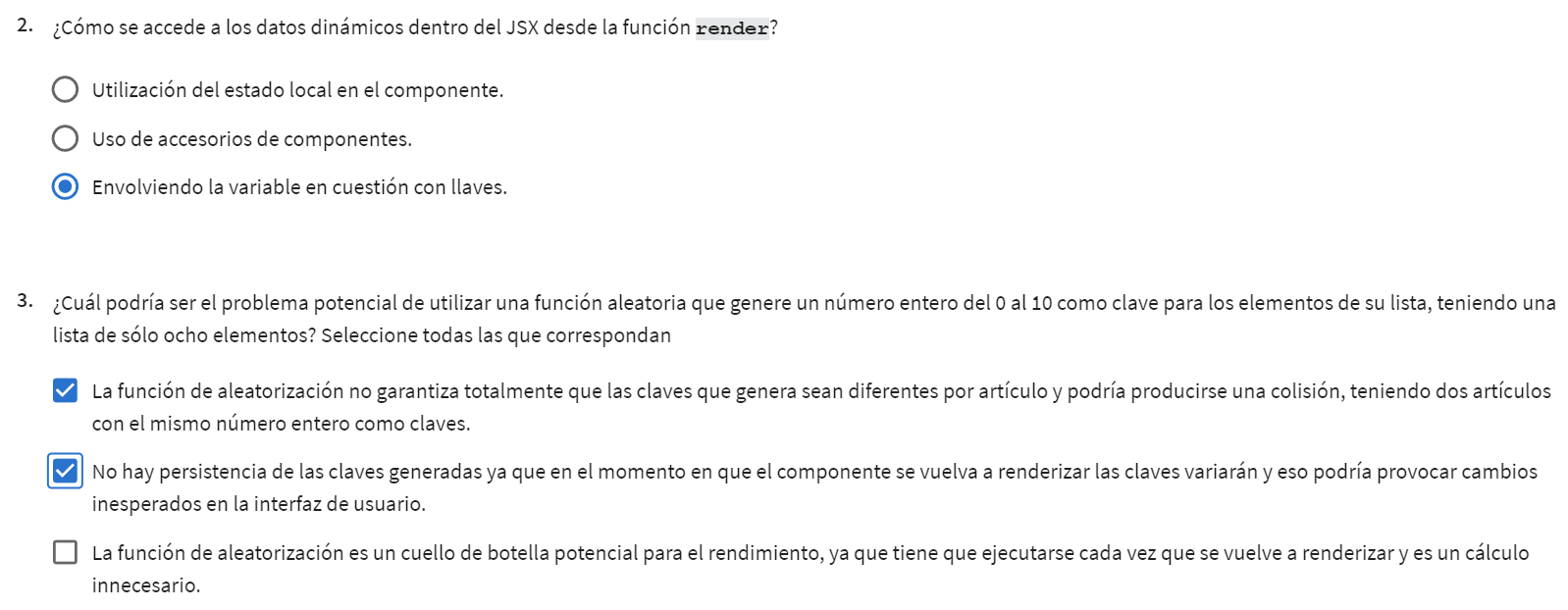
****

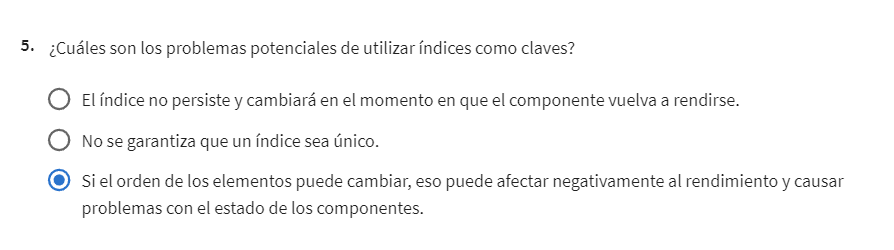














<https://react.dev/learn/rendering-lists#where-to-get-your-key>

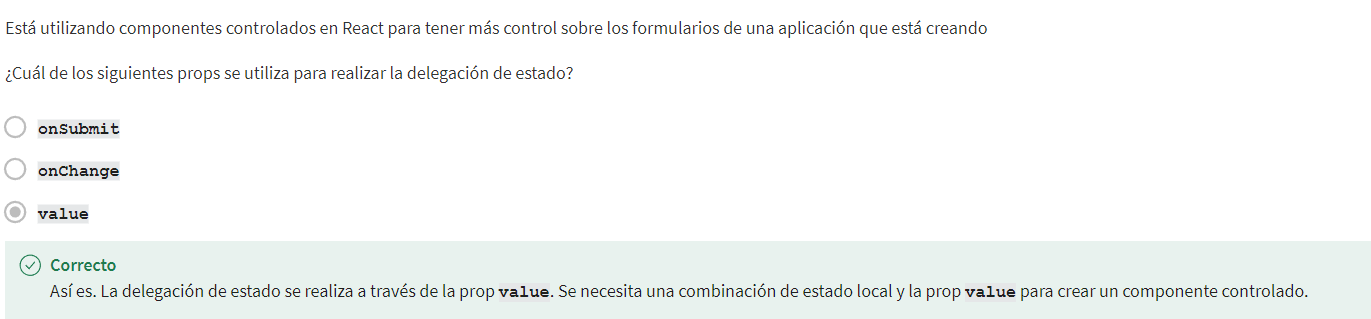
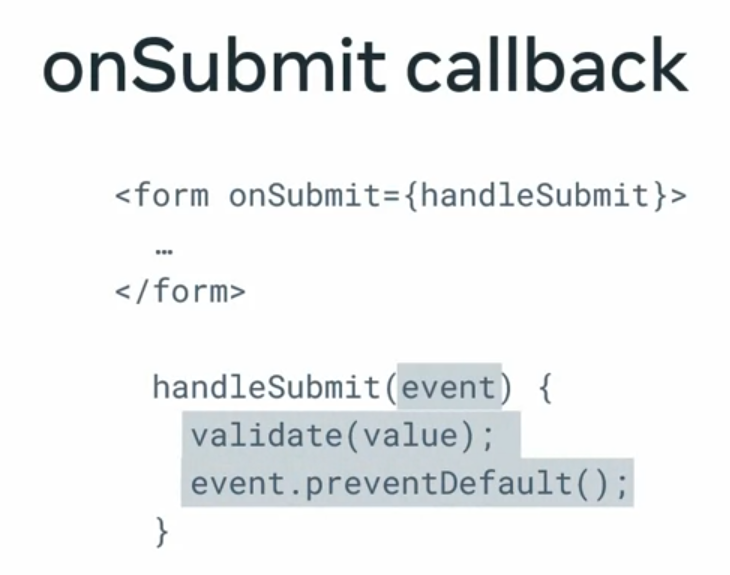
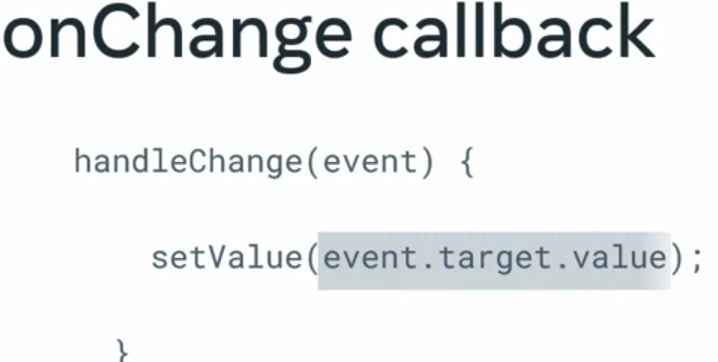
<https://react.dev/learn/rendering-lists#rendering-data-from-arrays>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map>

**Componentes controlados**

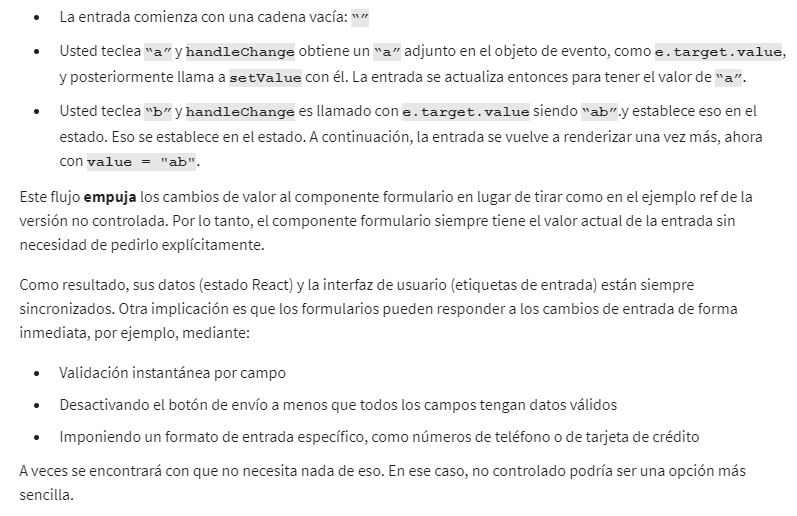
Para crear un componente controlado, necesita utilizar una combinación de estado local y la propiedad value. Inicialmente, asignará el estado local a la propiedad value.

¿Pero cómo obtiene actualizaciones de cualquier nuevo carácter de texto introducido en la entrada? Pues para eso, necesita una segunda propiedad para completar el diseño de su componente controlado, la llamada de retorno onChange. La llamada de retorno onChange recibe un parámetro de evento, que es un objeto de evento que representa la acción que acaba de ocurrir, similar a los eventos de los elementos DOM. Para obtener el nuevo valor de cada pulsación de tecla, es necesario acceder a la propiedad target del evento y tomar el valor de ese objeto, que es una cadena. Por último, para tener control sobre los valores del formulario cada vez que se envía el formulario, se puede utilizar la propiedad onSubmit en el elemento HTML del formulario.

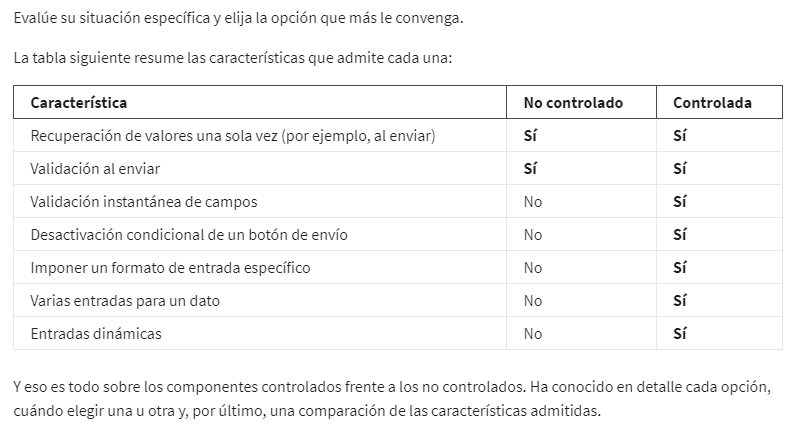


**Componentes controlados frente a componentes no controlados**

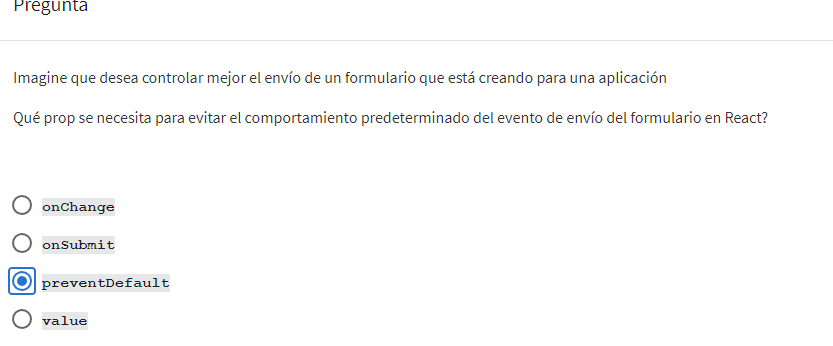


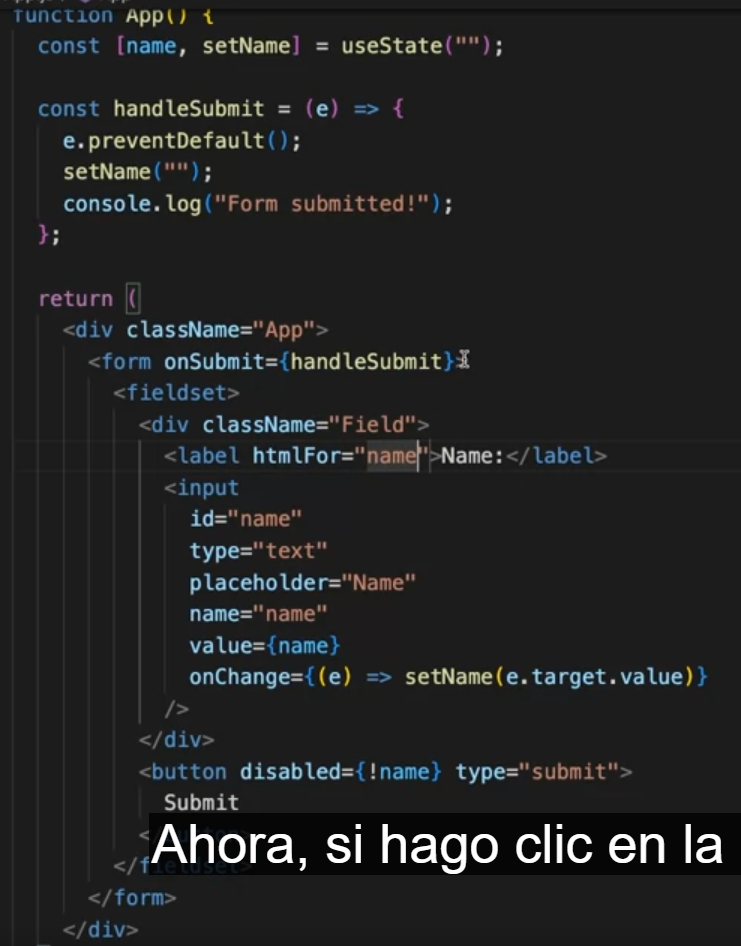


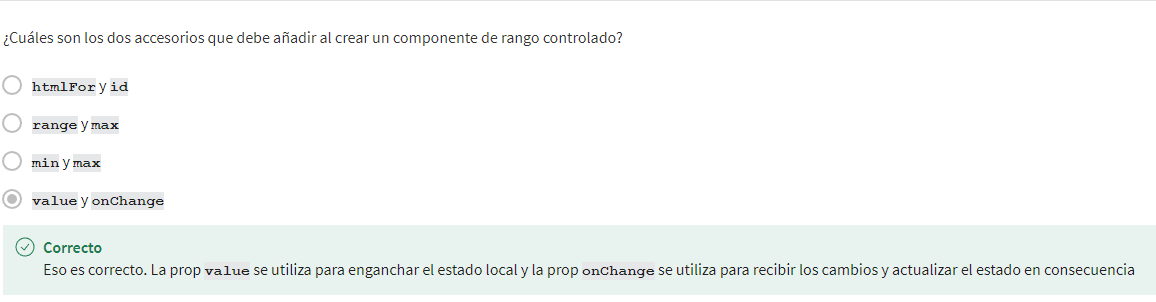
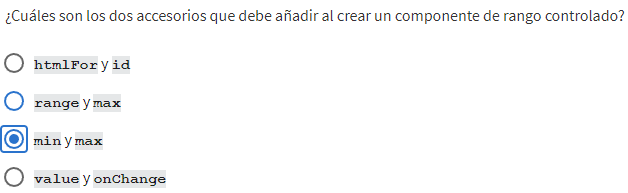


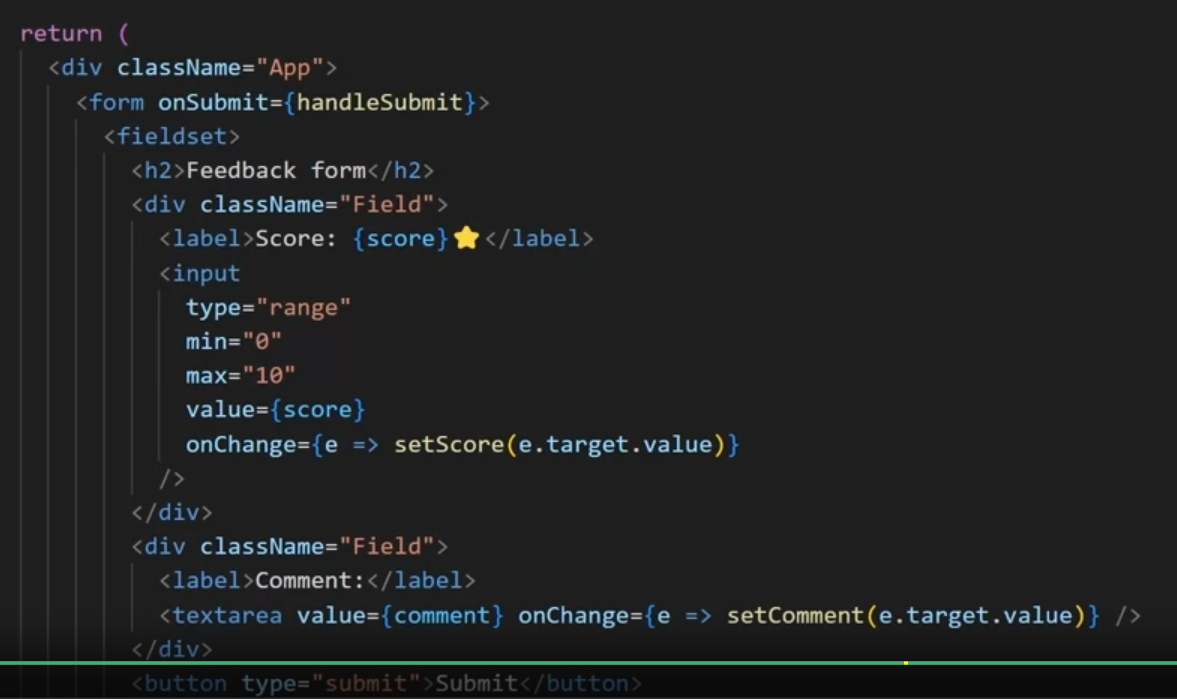
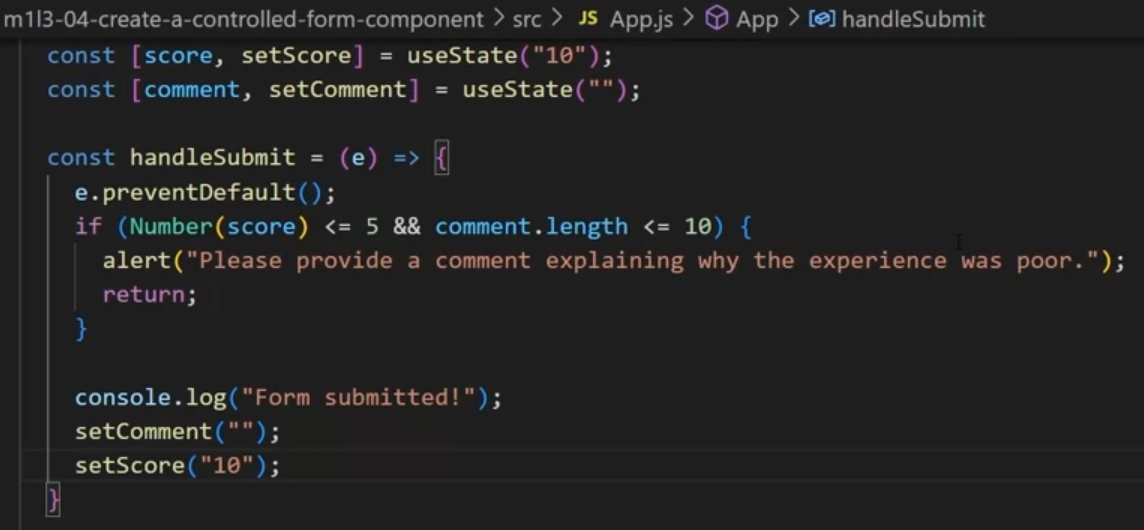
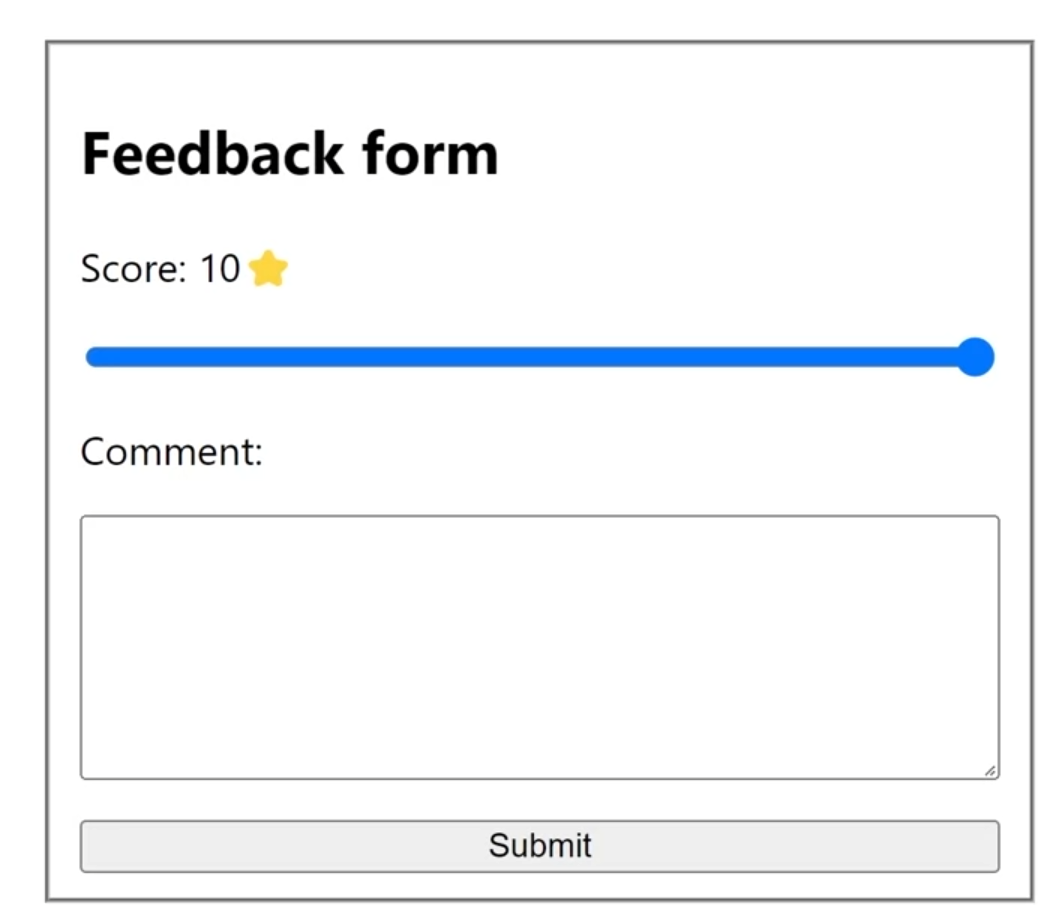


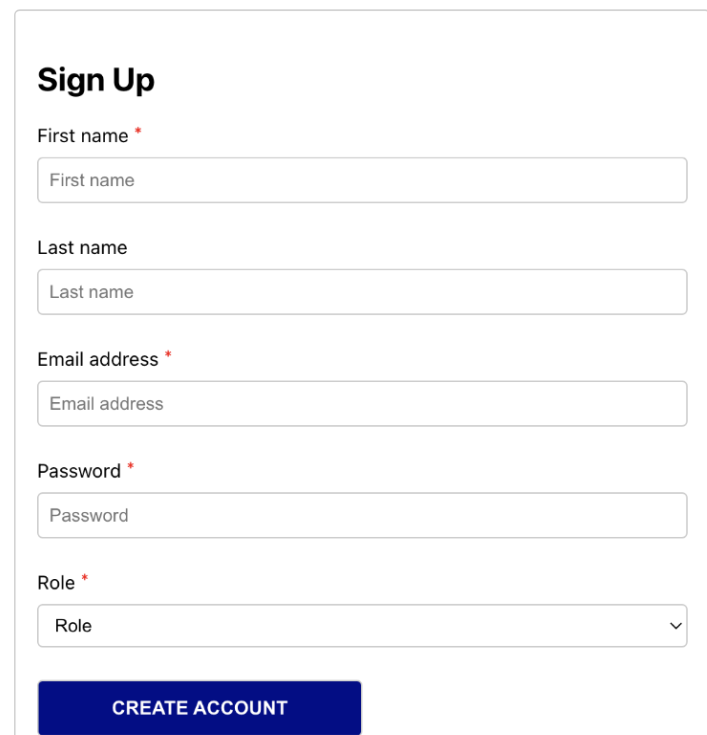
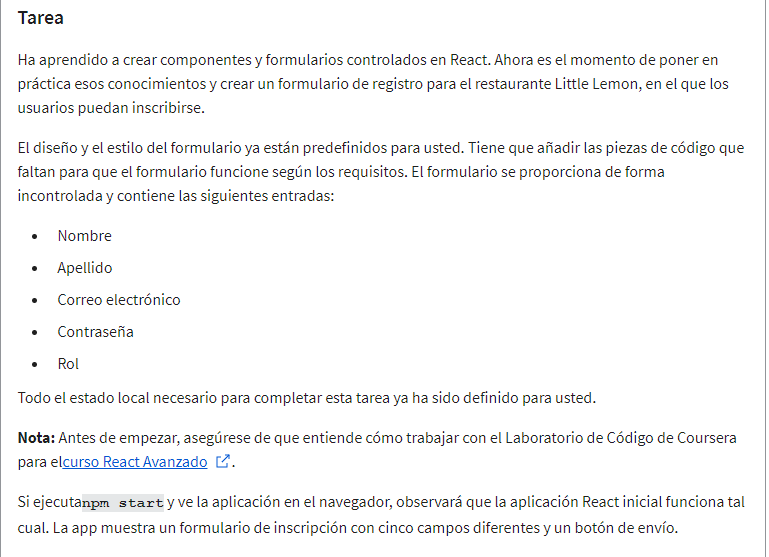
Recuerde que los componentes controlados en React son aquellos en los que los datos del formulario son gestionados por el estado del componente. Mientras que los componentes no controlados son aquellos en los que los datos del formulario son manejados por el propio DOM.

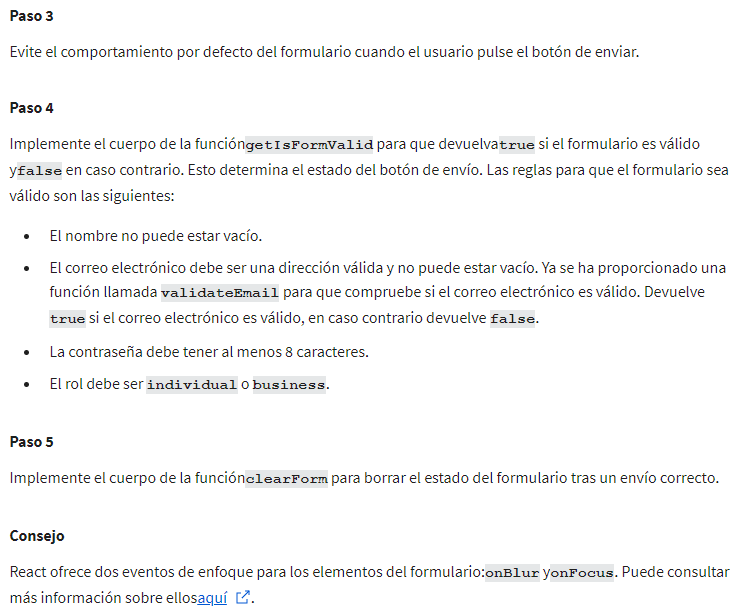
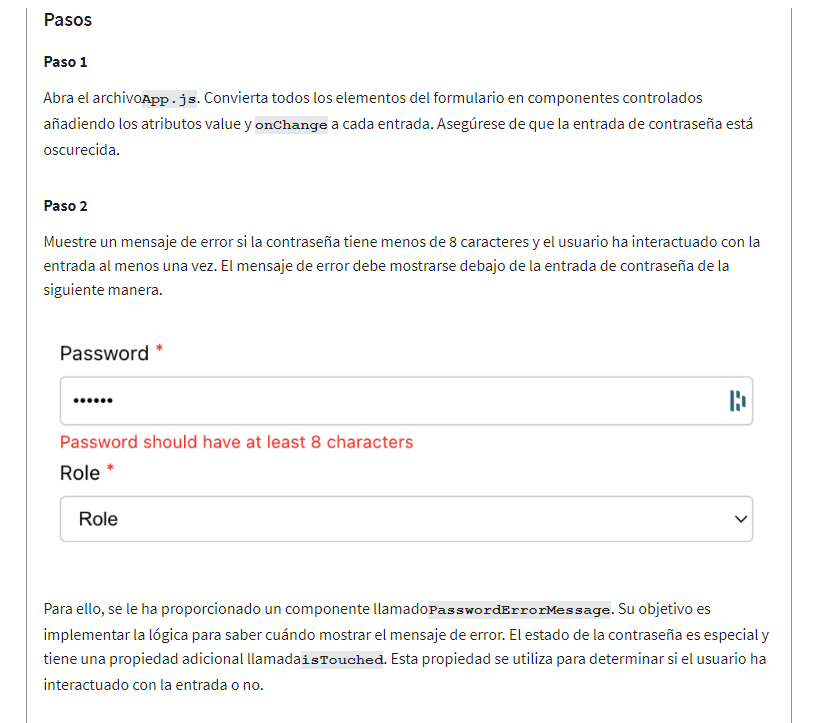






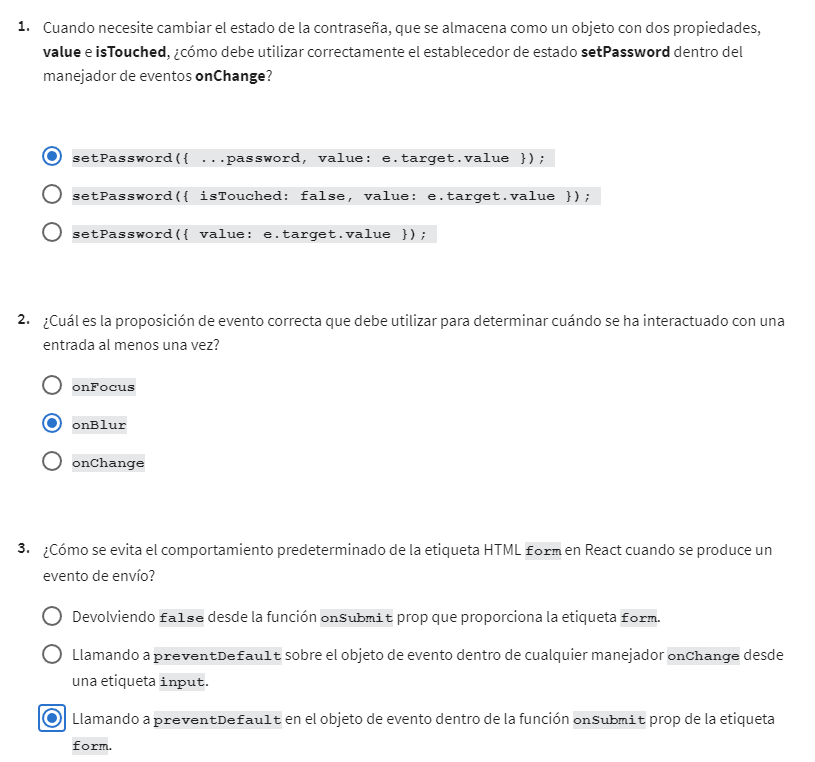


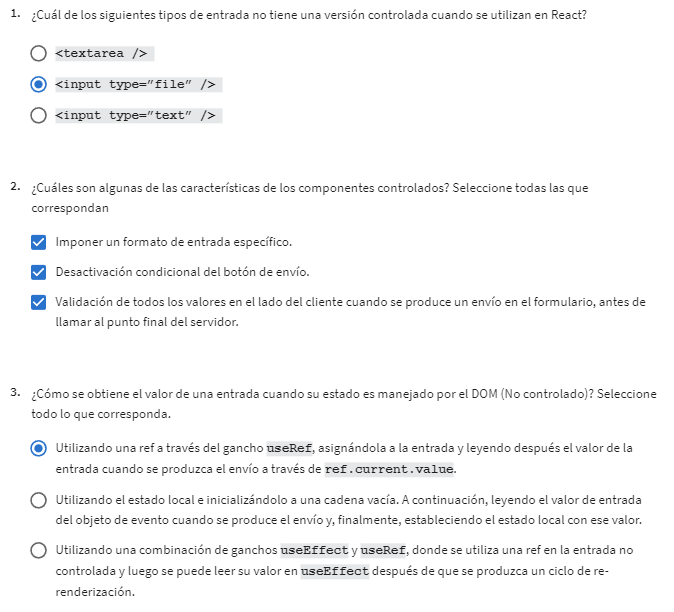


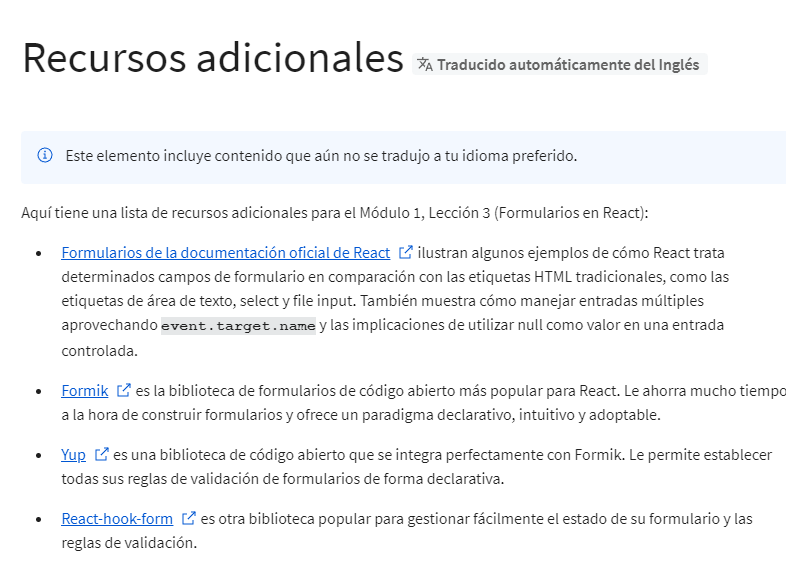


<https://legacy.reactjs.org/docs/events.html#focus-events>









<https://legacy.reactjs.org/docs/forms.html>

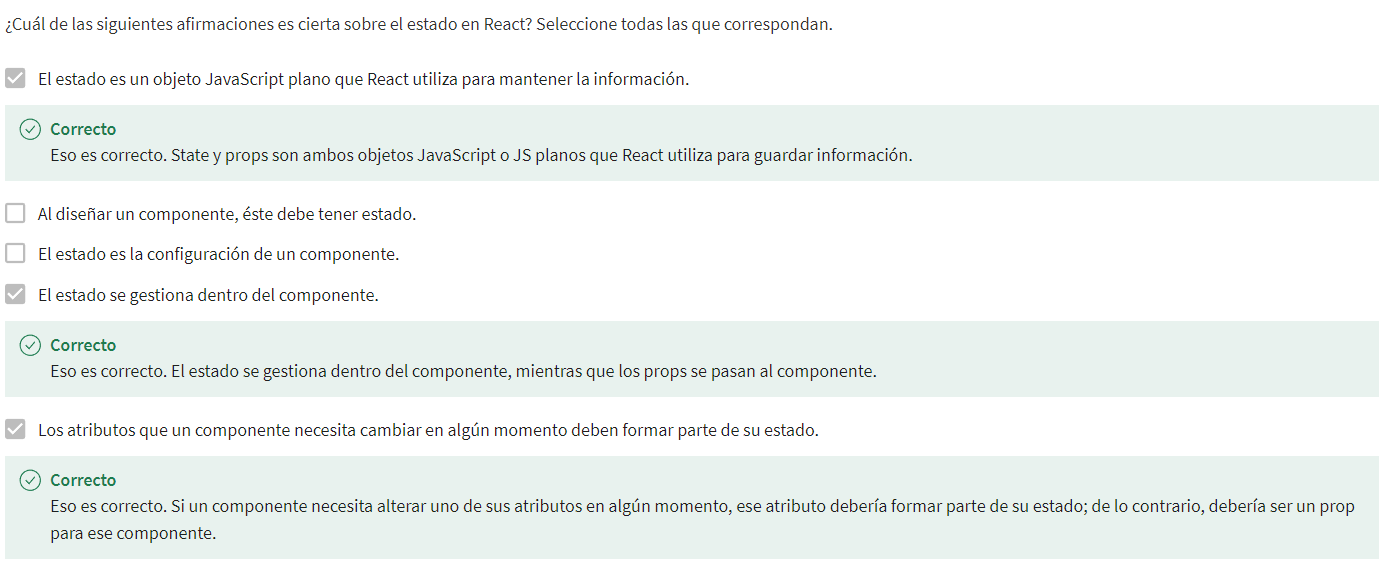
<https://formik.org/>

<https://github.com/jquense/yup>

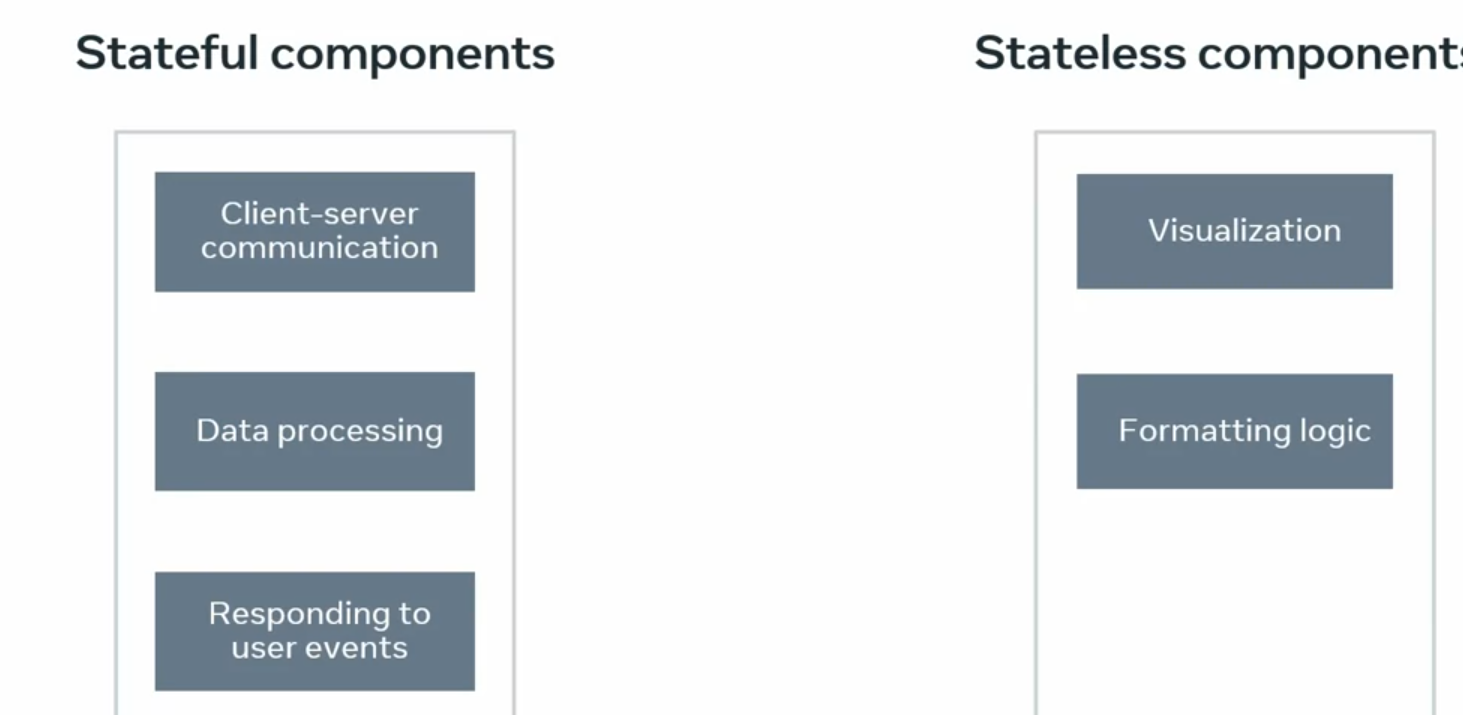
<https://github.com/react-hook-form/react-hook-form>



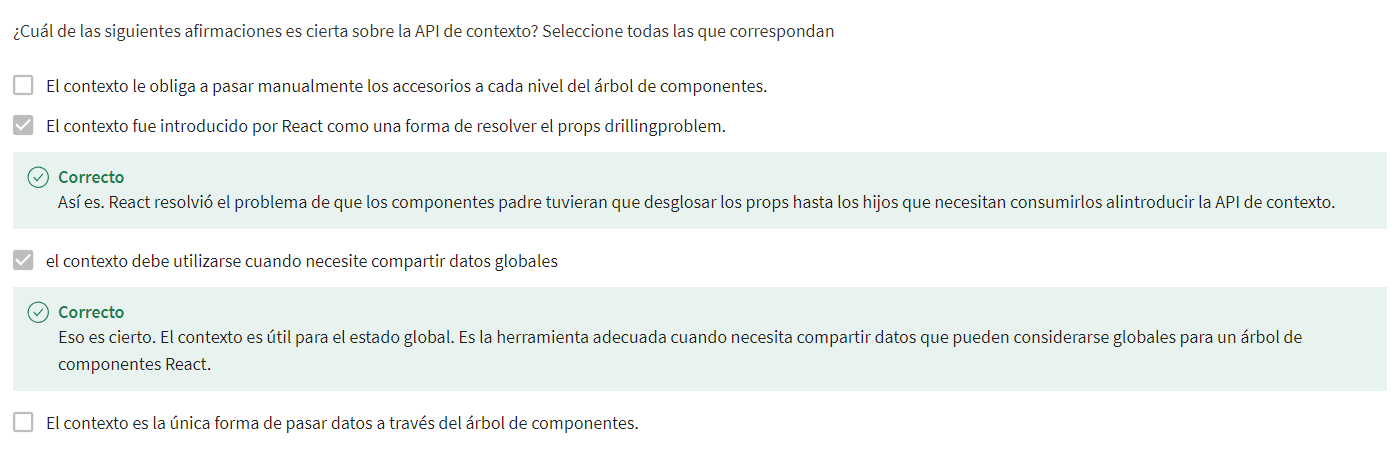
¿Cómo sabe si algo va dentro de props o state? La regla general es que si un componente necesita alterar uno de sus atributos en algún momento, ese atributo debería formar parte de su estado. De lo contrario, debería ser simplemente una prop para ese componente.

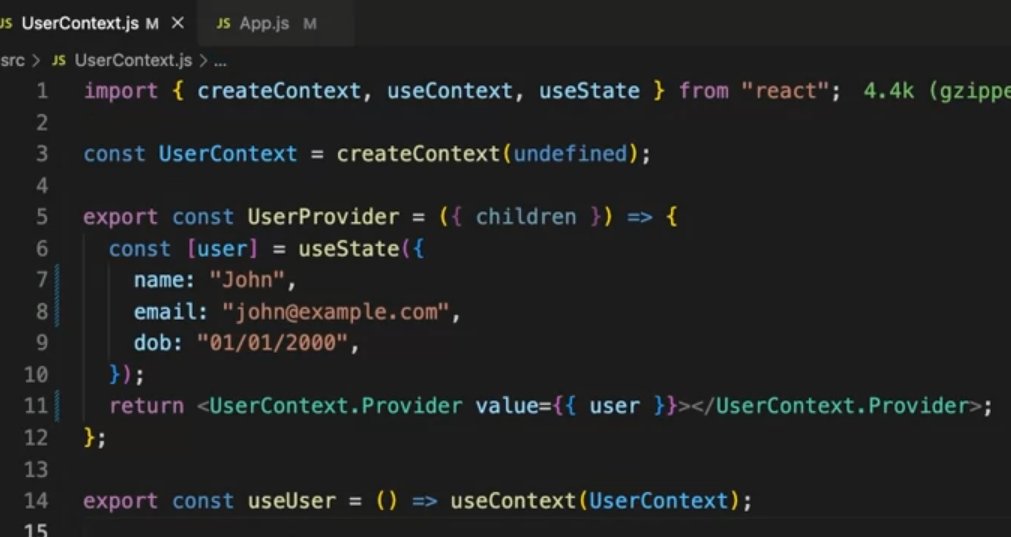


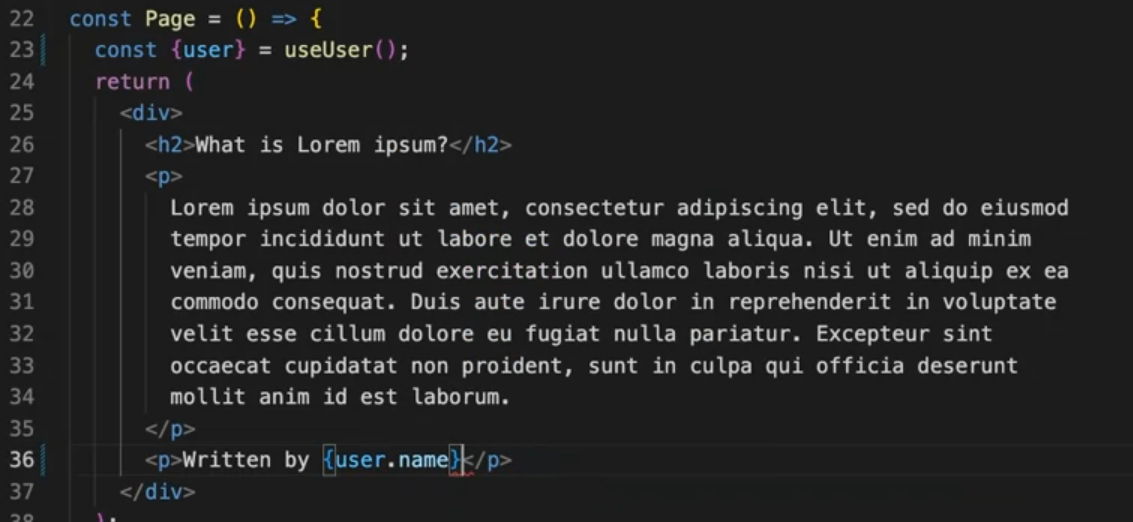
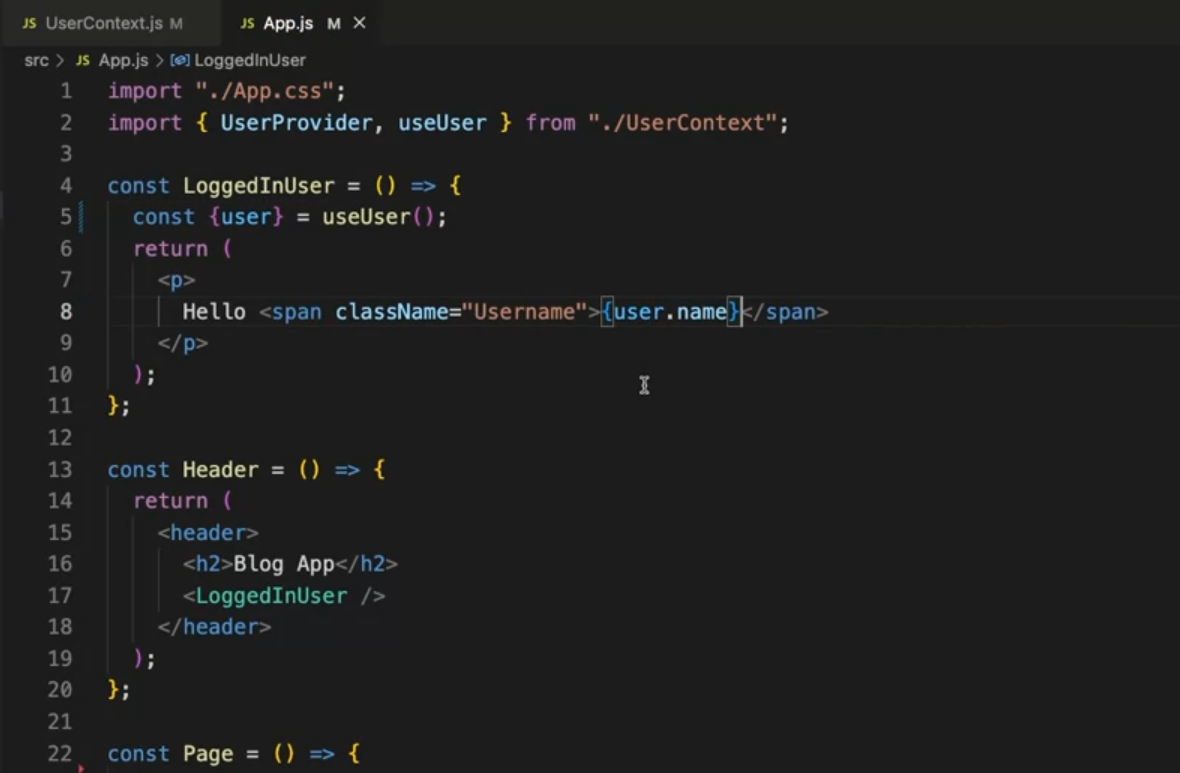
tipos de componentes react basados en props y componentes con estado pueden ser stateless o stateful. Los componentes sin estado sólo tienen props y ningún estado no hay mucho que hacer aparte de la función render y toda su lógica gira en torno a los props que reciben. Esto los hace muy fáciles de seguir y probar por otro lado, los componentes con estado tienen tanto props como estado. Se encargan del servidor cliente, la comunicación, el procesamiento de datos y de responder a los eventos del usuario. Este tipo de logística debería encapsularse en un número moderado de componentes con estado. Mientras que toda la lógica de visualización y formateo debería moverse hacia abajo en tantos componentes sin estado como sea posible.

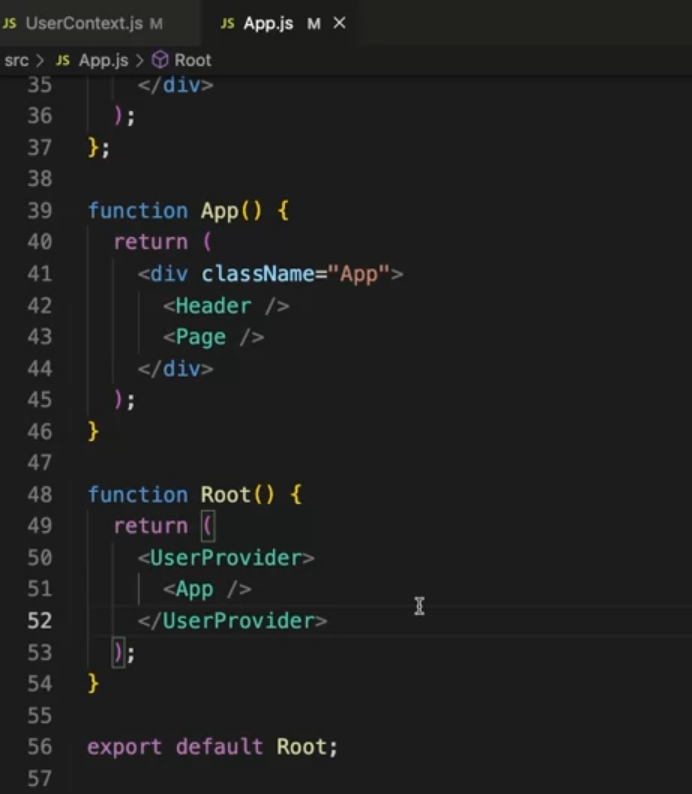


¿Qué es el contexto y por qué se utiliza?

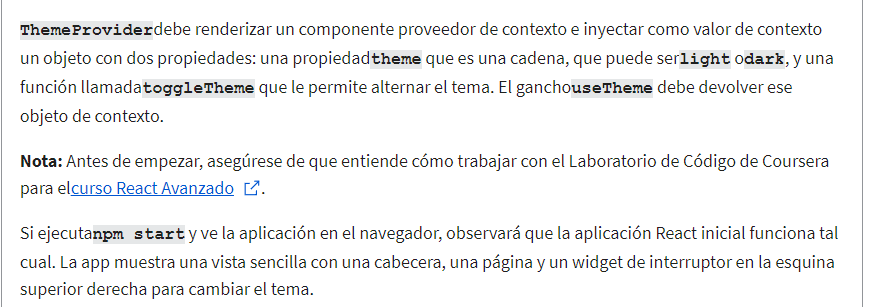


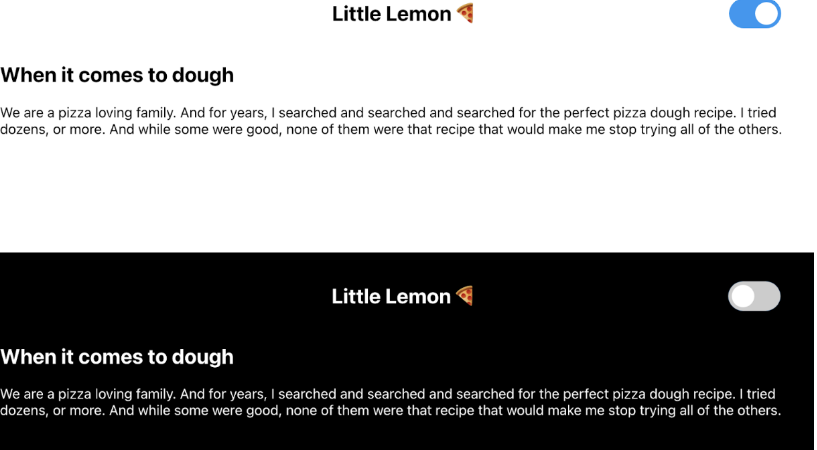
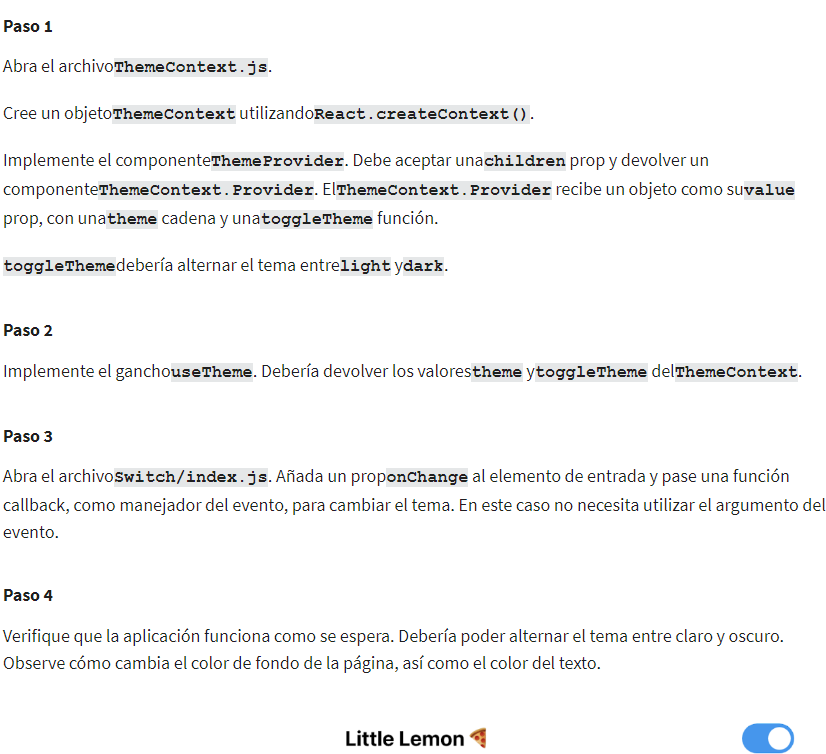


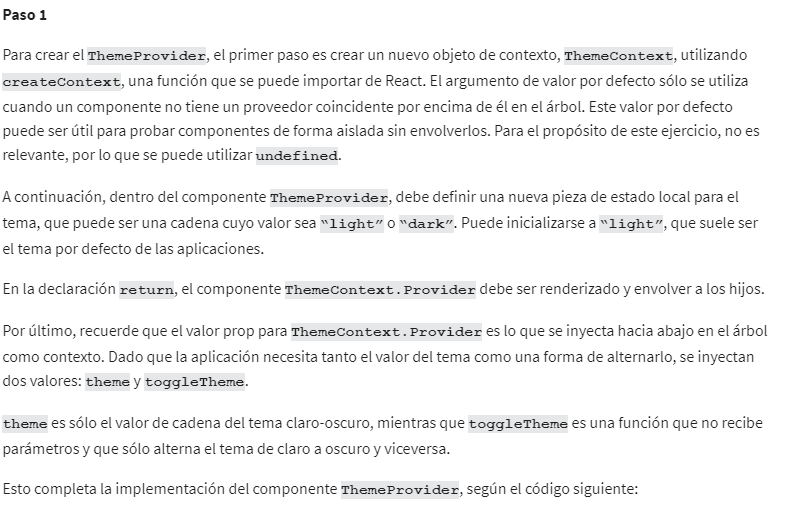


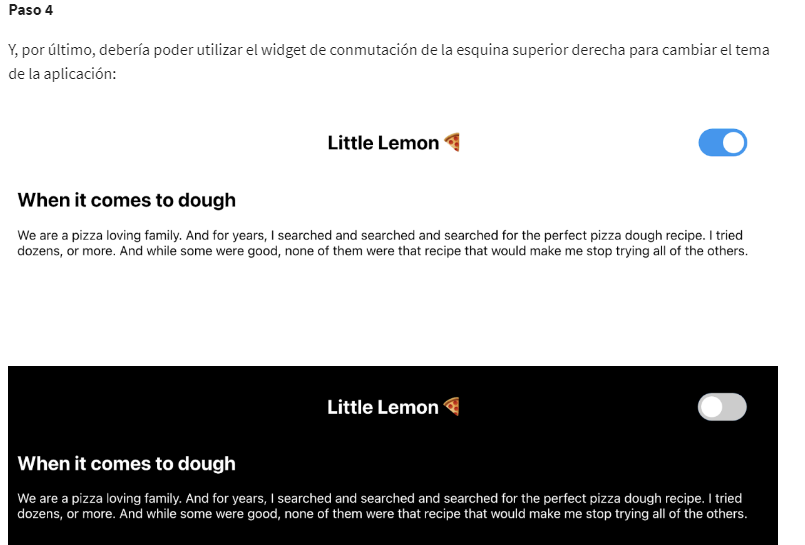
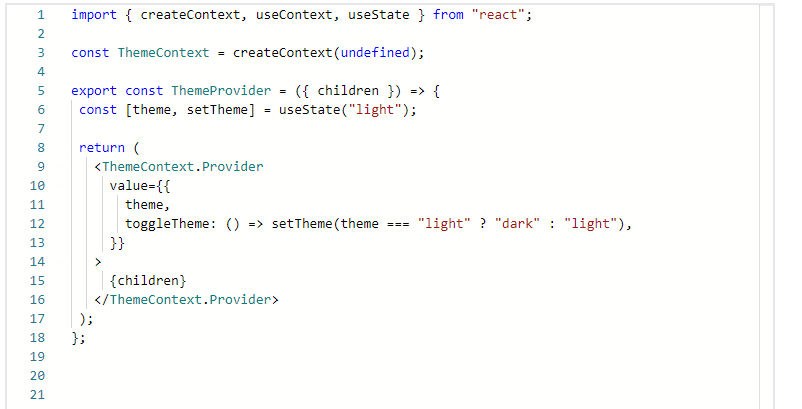


**Ejercicio: Crear un conmutador de temas claro-oscuro**

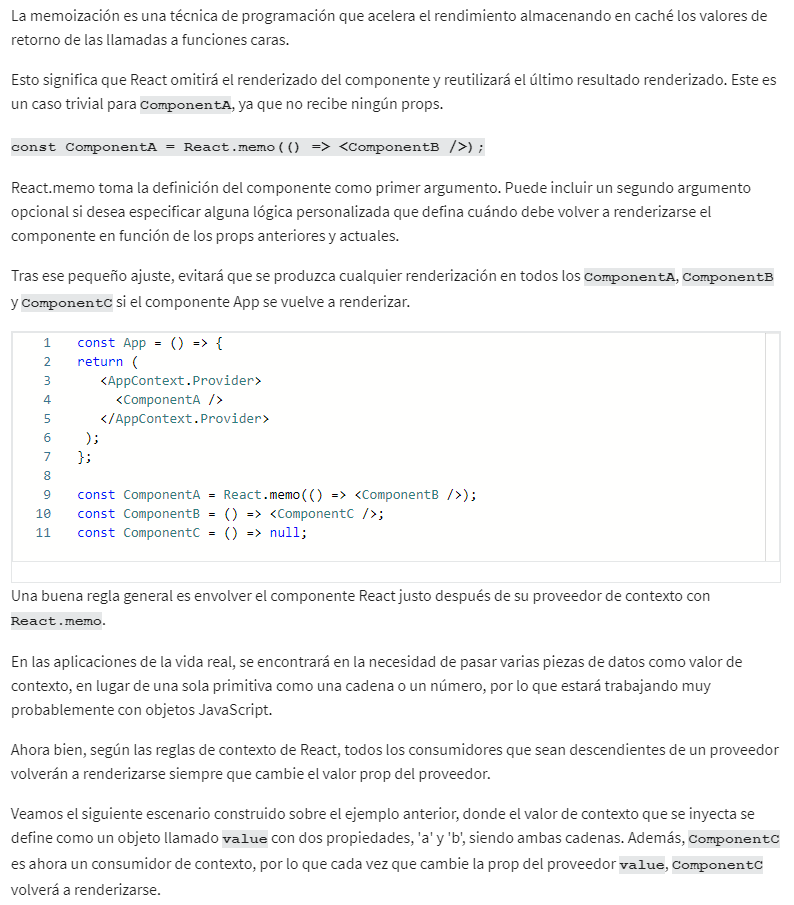
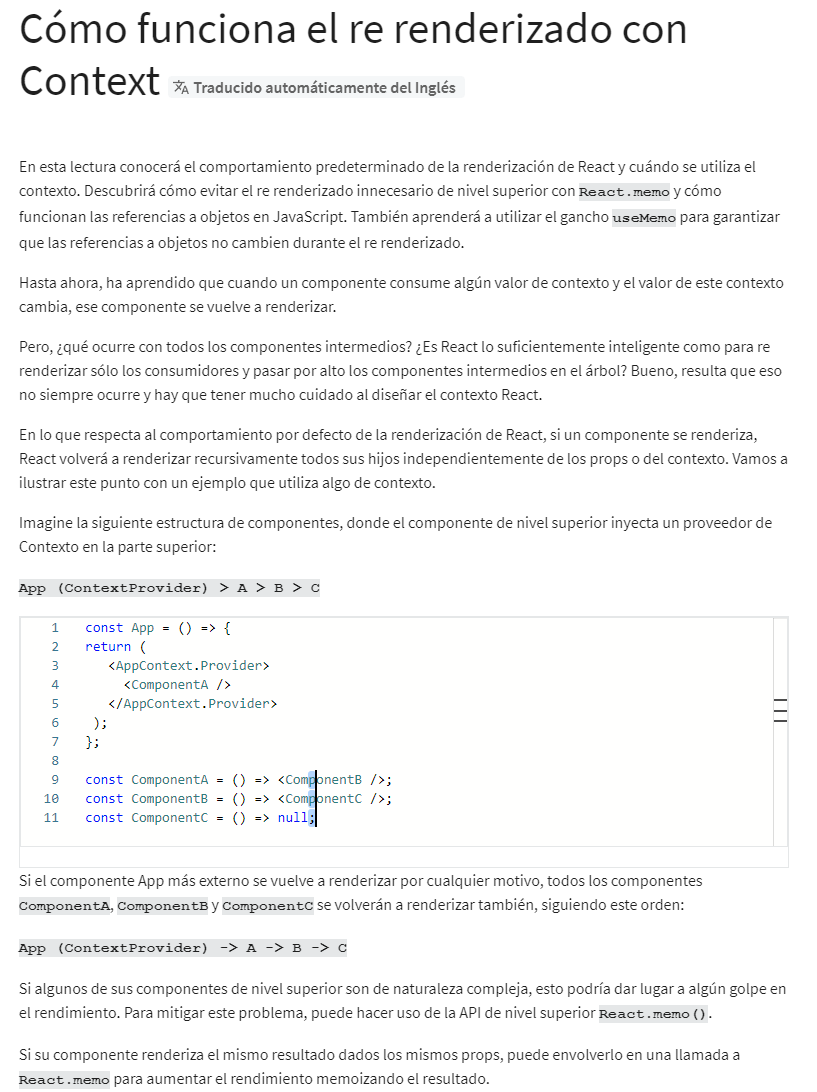


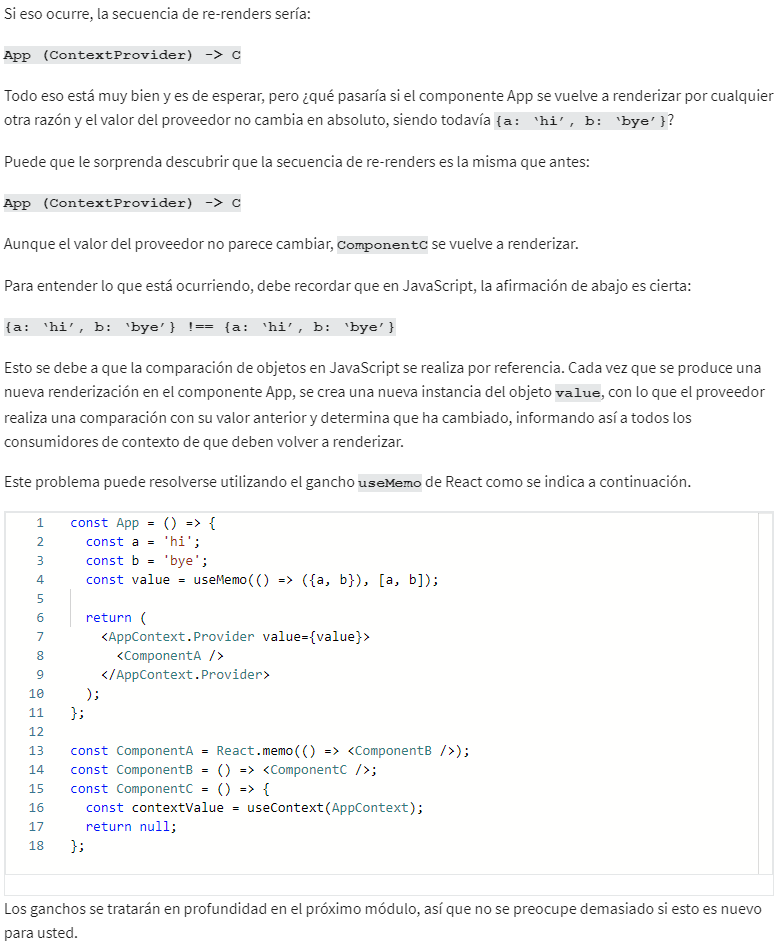


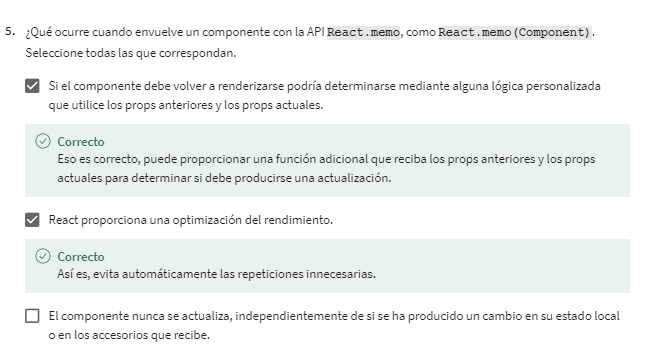
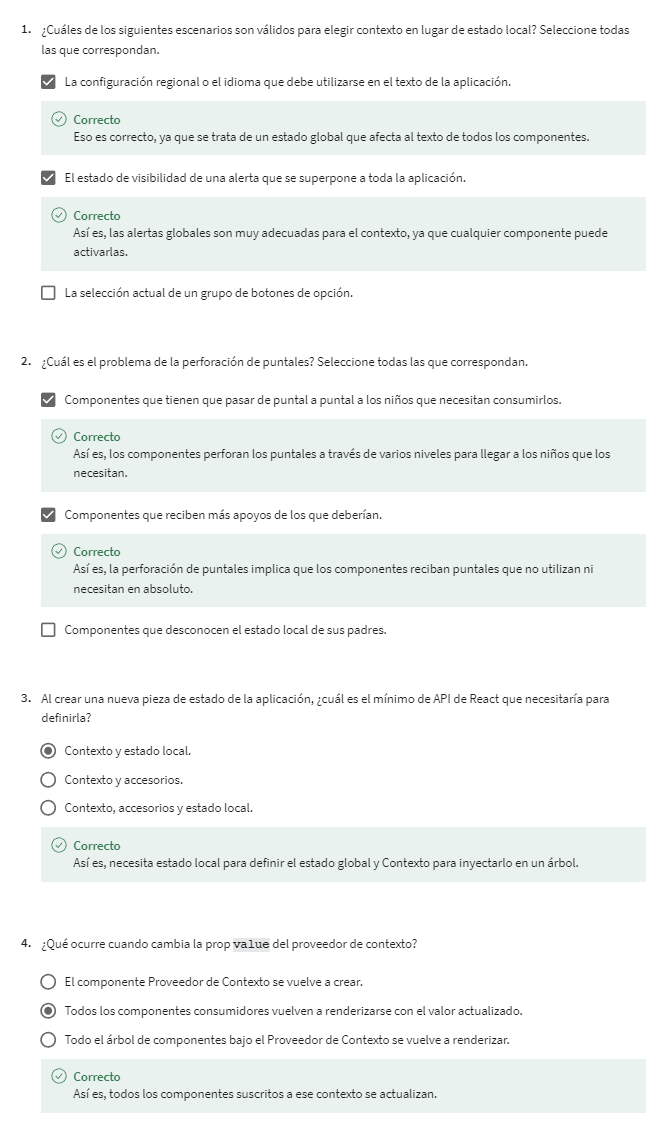
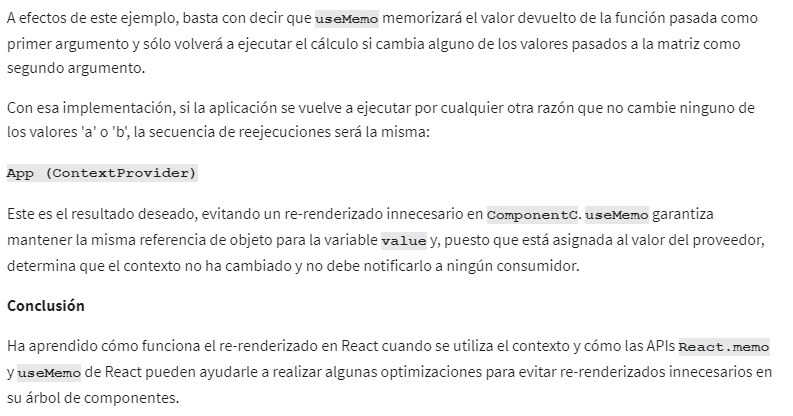


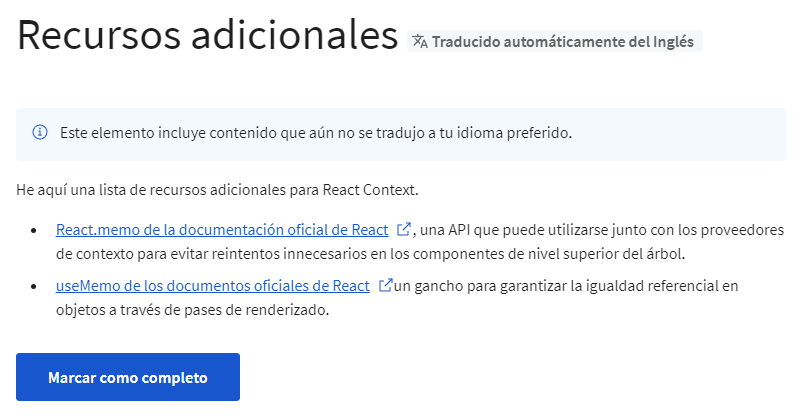
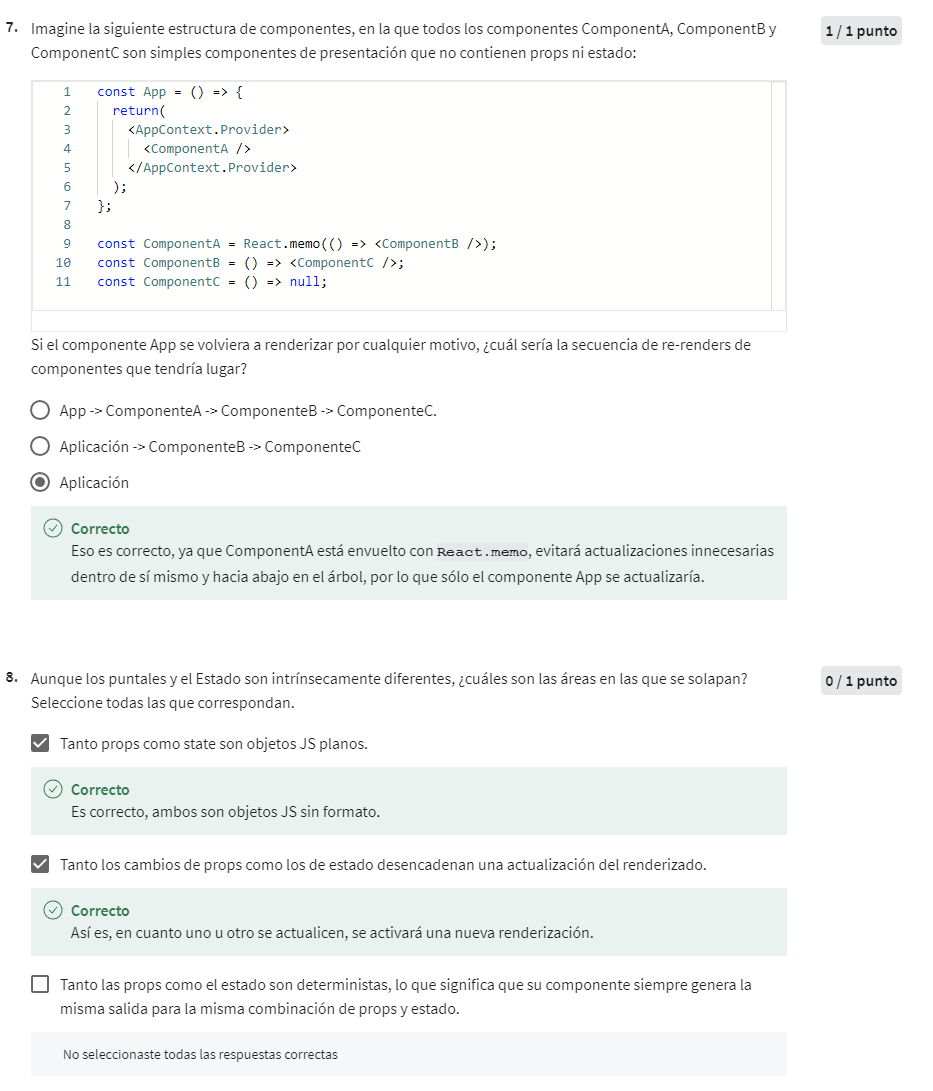
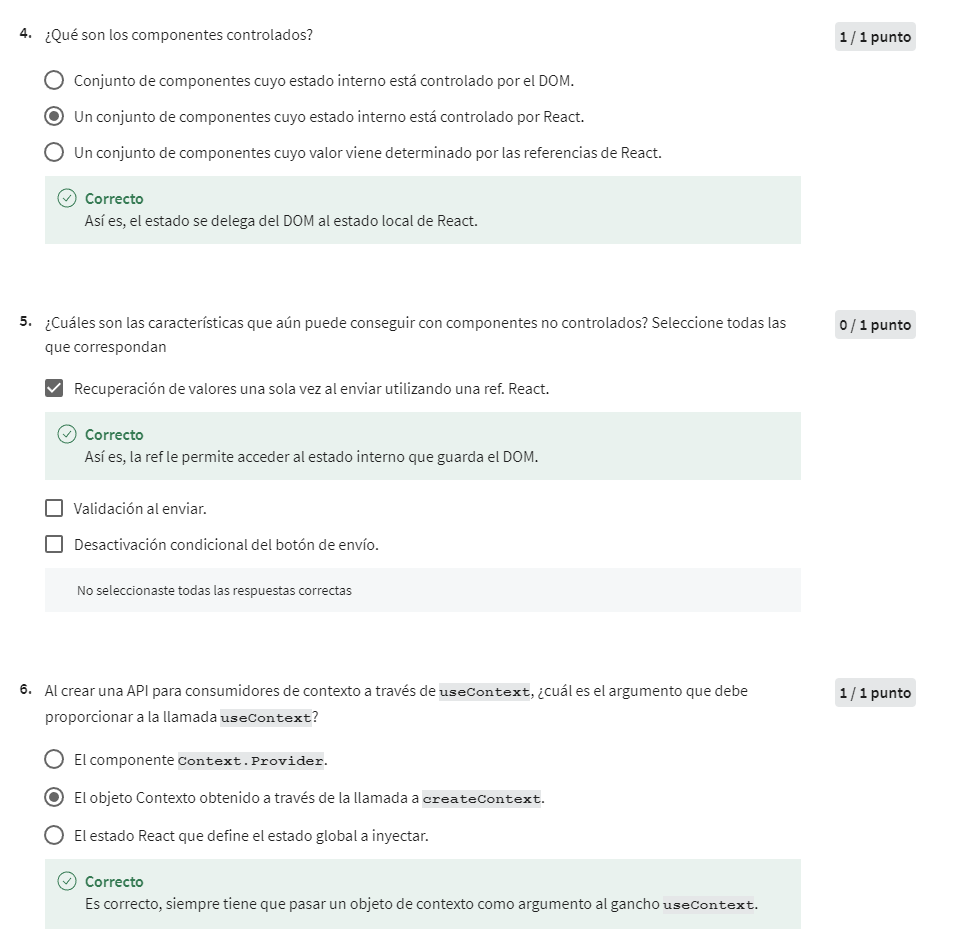
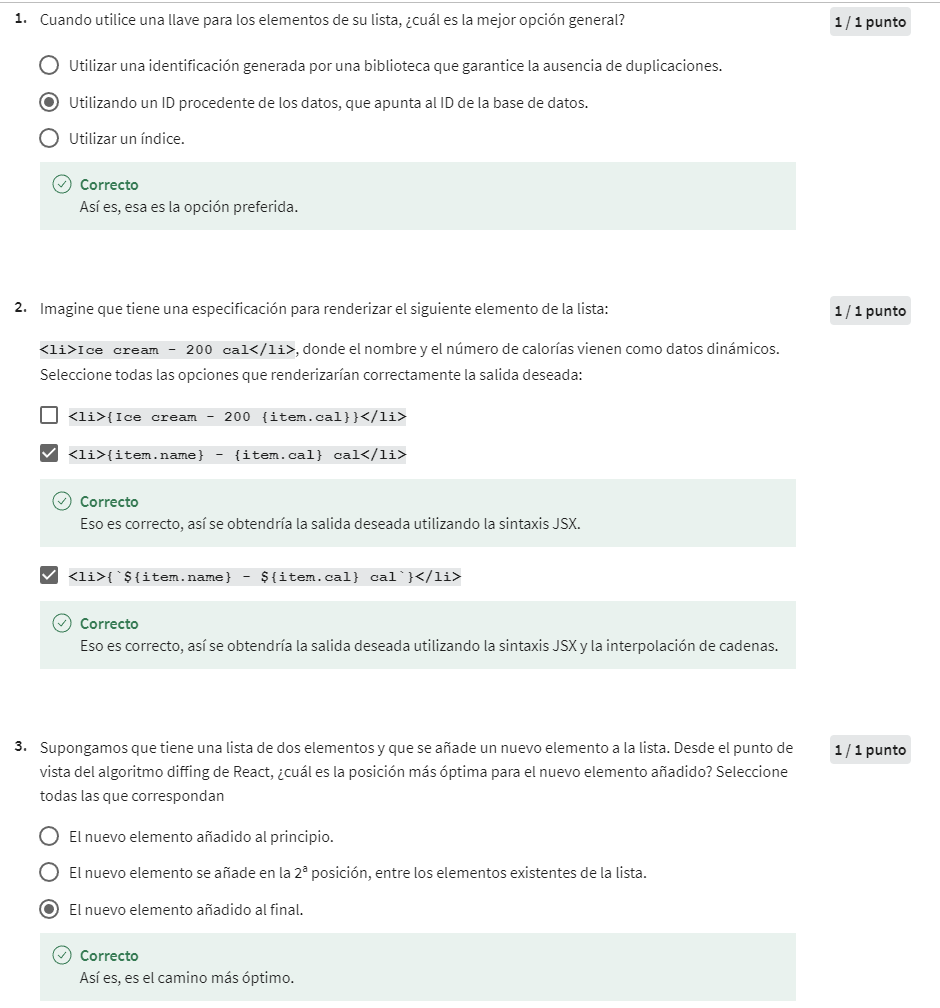












<https://legacy.reactjs.org/docs/react-api.html#reactmemo>

<https://legacy.reactjs.org/docs/hooks-reference.html#usememo>

