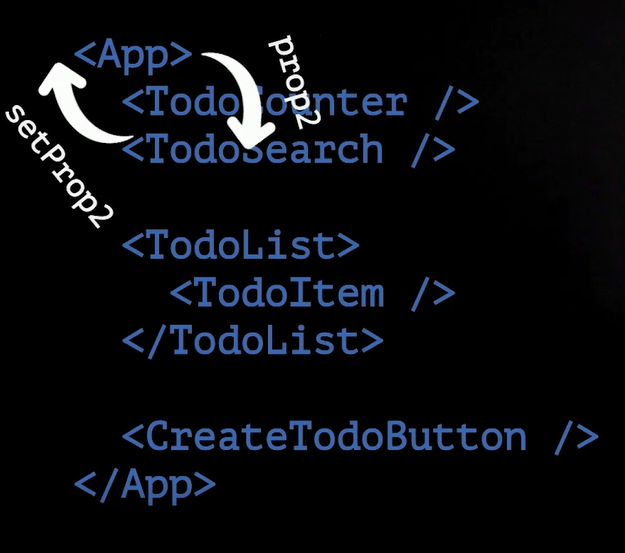
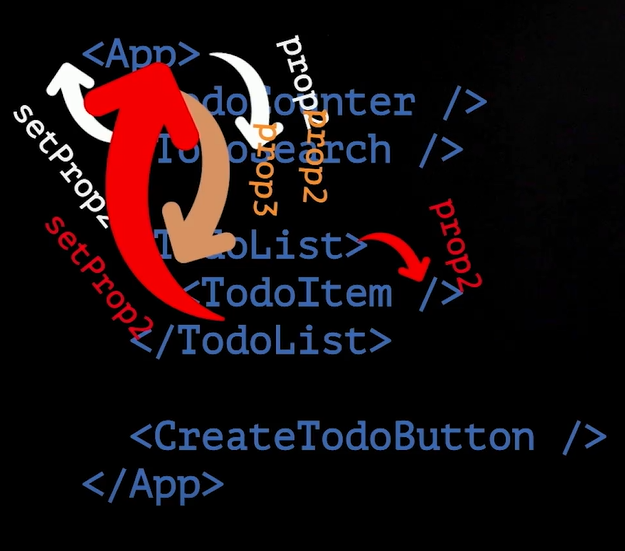
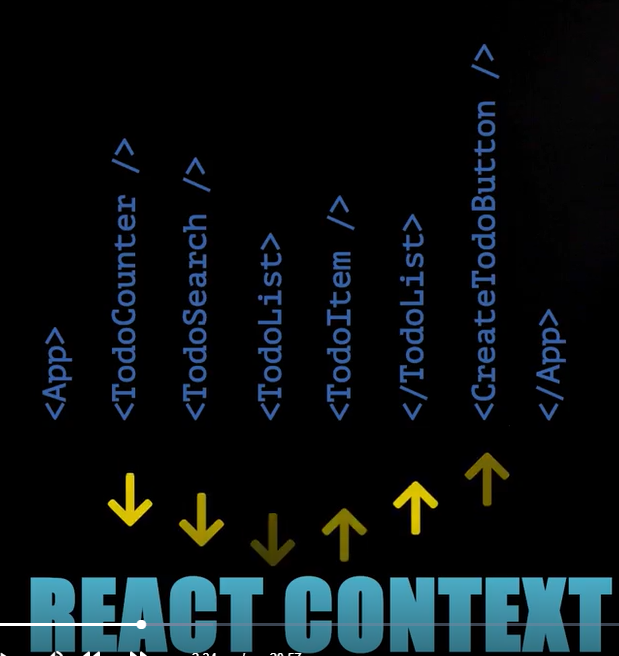
React context, nos permite crear estados globales para evitar el problema de prop drilling en las apps.

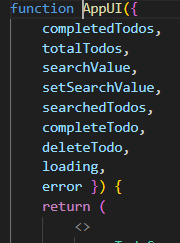
Como los componentes se comunican por medio de props, y hay unas propiedades que actualizan el estado, y por ende cambio los props que se comparten entre componentes

 Por ejemplo todo search o counter, puede enviar actualizaciones a componente todo App para que este componente actualice una propiedad y esa propiedad no solamente actualiza ese componente que la actualizo, sino que también a otros componentes por el ejemplo el todo list, entonces el todo list se las pasa al todo item, este lo recibe y se las vuelve a actualizar al todo App

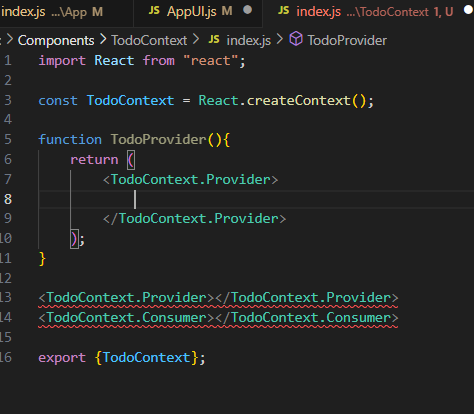
 Esto significa peligro, esto es prop drilling.

Para esto no vamos a mirar los componentes como una estructura de Árbol si no como componente y ya, sin importar el nivel en el que estén, sin importar sin padres o hijos o abuelos, son componentes y ya, y lo que permite React conext, nuestro estados generales, es que todos los componentes puedas comunicarse directamente con el estado, con CONTEXTO GLOBAL de la aplicación y de esta forma no se va a tener que entre componentes se pasen información, si no que cada componente que necesite informaciones la pide al contexto o estado general.



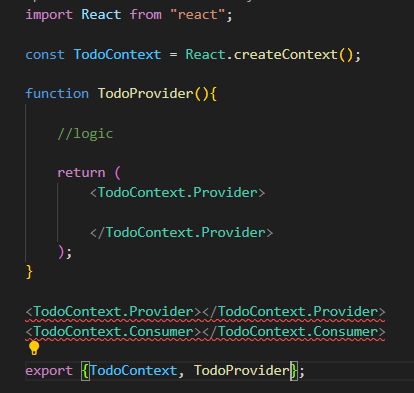
 A screenshot of a computer program

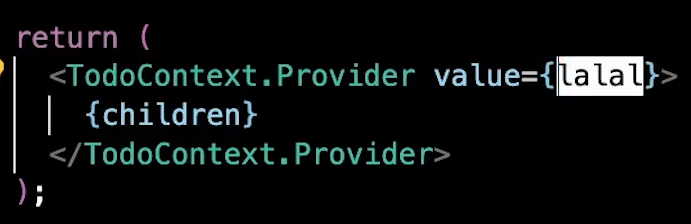
Description automatically generated



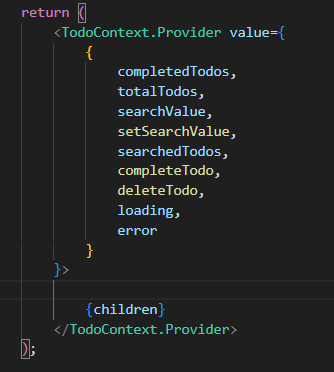
Para ello se crear un componente para el context, para ello se usa el createContext, y al tener este un puede acceder al Provider y consumer, se puede usar directamente, pero se crea el fuction para tener un proveedor personalizado. Esto da los siguientes beneficios:

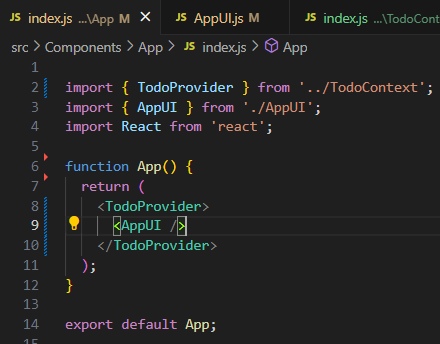
* Nombre más cómodo
* Se exportar el TodoProvider como tal
* Dentro del fuction se puede encapsular la lógica que se quiere compartir entre varios niveles de los componentes

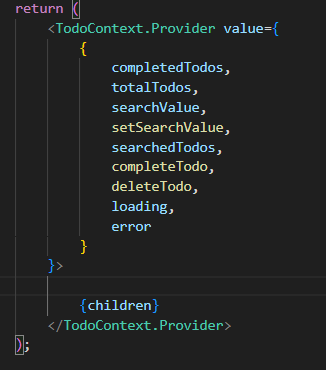




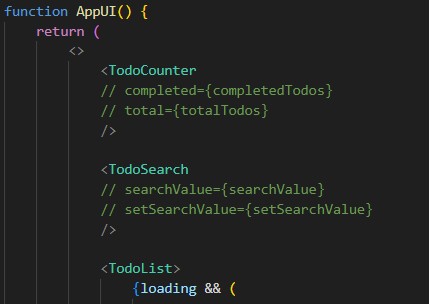
Todo lo que este acá es lo que vamos poder exponer globalmente en el contexto de la aplicación

 ahora con esto, cualquier componente que va este dentro puede acceder a las propiedades dentro del value

 con esto se manda el AppUI como el children, de esta manera el componente AppUI y todos los componentes hijos van a poder acceso a la informaciones la propiedades en el contexto

 es decir, estas.

Ahora para usarlo en el APPUI



A screen shot of a computer

Description automatically generated Se usa de la siguiente, manera pero este Todo.Context.cosumer no espera que le enviemos directamente un componente si no un reder fuction



De esta manera, es decir un return dentro de otro return, se usa de esta manera, ya que vamos a recibir todas propiedades que necesitamos, como parámetros

A screen shot of a computer program

Description automatically generated

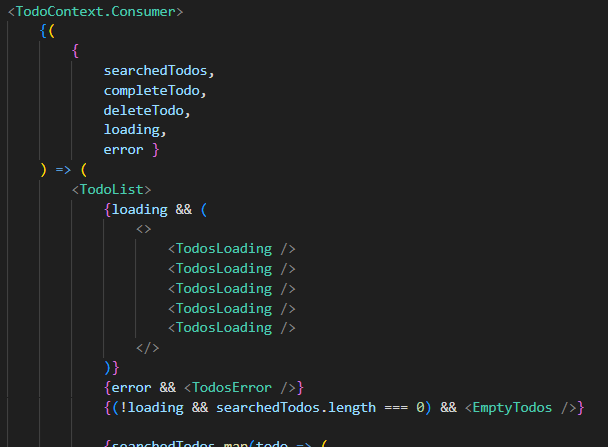
**Cualquiera información general de la aplicación puede ser un context: TODOS, usuarios, validaciones, etc.**

A computer screen shot of a program code

Description automatically generated

En vez de usar el TodoContext.consume,

Se puede usar el UseContext, el cual nos permite hacer lo mismo de una forma mas sencilla



A esto  con el useContext

Apuntes:

Recién tuve una entrevista con esa información. Es muy importante ver los términos ya que es algo que se ve comúnmente en react:

Prop Drilling: es una paso del desarrollo que ocurre cuando necesitamos obtener datos que están en varias capas en el árbol de componentes React

Este se ve siempre cuando pasamos props entre hijos, luego ese a otros hijos y así sucesivamente... la solución es usar context:

Context: es un elemento que podemos crear en React para establecer una comunicación directa entre un componente en un nivel muy alto y uno en un nivel mucho más bajo.

Por ende context permite acceder a los valores de forma directa. React Context es como una nube donde puedes almacenar funcionalidades, estados, datos, etc. Que necesitan ser compartidos entre diferentes componentes de una aplicación. Digamos que existen varios componentes que necesitan acceder a un mismo estado, como por ejemplo, un estado de carga en lugar de pasar estos datos a través de múltiples niveles de componentes, que conducen al Prop Drilling siendo un patrón que puede ser útil en algunas situaciones, pero también puede resultar incómodo y difícil de mantener si hay muchos niveles de componentes intermedios involucrados. Además, puede hacer que el código sea menos legible y más propenso a errores.

Por ello React Context es una buena solución y acceder a ellos fácilmente desde cualquier lugar de la aplicación. Por ello, otra manera de verlo es como una biblioteca compartida. Imagina que trabajas en una empresa y necesitas compartir una biblioteca de recursos (libros, revistas, etc.) con tus compañeros de trabajo. En lugar de pasar los libros de mano en mano entre los compañeros de trabajo, puedes ponerlos todos en una biblioteca compartida en el edificio de la empresa o en englobarlos en un Google Drive, lo que hace que sea más fácil para todos acceder a los recursos que necesitan.

De manera similar, con React Context, puedes guardar datos que necesitan ser compartidos entre diferentes componentes de tu aplicación en una "biblioteca compartida" (el contexto de React), lo que facilita el acceso a estos datos desde cualquier lugar de tu aplicación sin tener que pasarlos manualmente a través de múltiples niveles de componentes.