El patrón Redux – Ventajas del patrón Redux

¿Por qué dedicar tiempo a aprender NgRx y el patrón Redux?

- Tener un árbol de estado inmutable centralizado hace que los cambios de estado sean más explícitos y predecibles con un rastro repetible de cambios de estado, lo que facilita la localización de problemas.
- El uso de funciones puras para cambiar el estado permite características en Redux como la depuración de tipo "Time Travel Debugging", "Record/Replay" y "Hot Reloading".
- También hace que sea fácil actualizar el estado de su aplicación desde el almacenamiento local o al hacer renderizado del lado del servidor.
- Redux facilita la implementación de una estrategia de "Change Detection" de Angular en sus componentes llamada "OnPush", que puede mejorar el rendimiento de sus vistas.
- El uso de Redux facilita la escritura de pruebas unitarias, ya que todos los cambios de estado pasan a través de funciones puras, que son mucho más simples de probar.
- Las herramientas son otro gran beneficio, ya que Redux hace posible tener un historial de los cambios de estado, lo que facilita la visualización de su árbol de estado, la depuración deshaciendo o rehaciendo los cambios de estado, mientras que también obtiene un log avanzado. Este es un gran cambio en la forma en que normalmente depuramos una aplicación Angular, y la mayoría de los desarrolladores se sorprenden de cuánto menos depuración tradicional hacen, como el console.log y el uso de breakpoints.
- Otro beneficio es la simplificación de la comunicación entre componentes. NgRx facilita el acceso al estado compartido mediante la inyección del Store en un componente en lugar de pasar datos entre ellos usando los inputs de los componentes.

Una frase que expresa el valor del patrón Redux es la siguiente:

"*Redux no es genial para hacer cosas simples rápidamente. Es genial para hacer simples cosas realmente difíciles*"