CURSO DE REDUX

React es una librería que nos ayudara hacer la tarea de crear interfaces.  
Tiene componentes que se crean una vez y se pueden usar miles de veces en donde tú quieras.  
Tiene tanta demanda porque tiene librería de peso ligero y su rendimiento es muy bueno. (Gracias a los creadores Facebook)  
JSX es la sintaxis con la que se escribe código en React (es una fusión de HTML y JavaScript).

**Stateful vs Stateless**

Los componentes no funcionales no manejan estado, solo manejan información y funciones.

Los componentes de clases manejan un estado interno, información y funciones, no hace falta definir las funciones con const simplemente con el nombre de la función ya lo detecta.

Stateful son componentes clases (no funcionales).

Stateless son componentes funcionales.

Las funcionales: manejan estados.

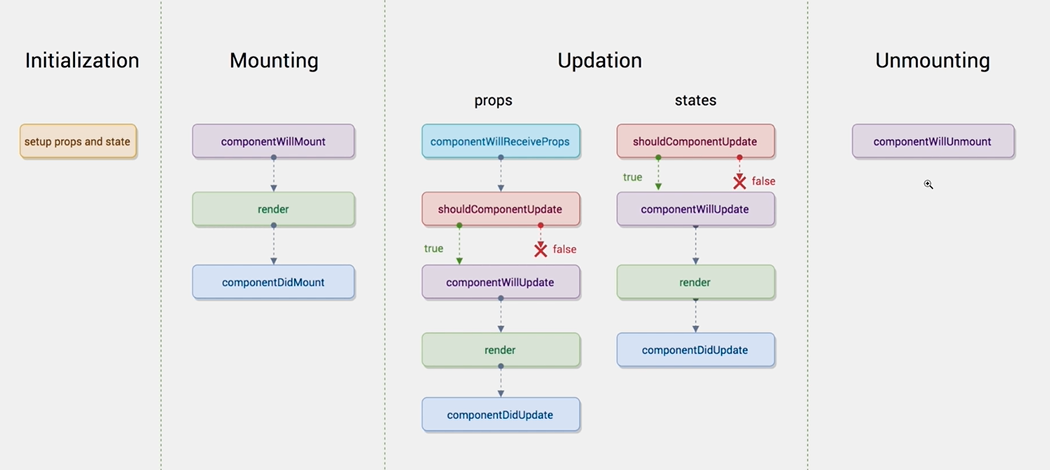
Las no funcionales: no manejan estado solo manejan información.

**Fases por las que un componente pasa**

**Inicialización** se declara nuestro estado o nuestras propiedades (setup props and state)-aparece siempre en el constructor.

**Montado** tres métodos en las cuales 1 es obligatorio que es el RENDER. El componentWillMount siempre se ejecutará antes que el render y el componentDidMount se ejecutará después del render.

**Actualización o editar** hay varias funciones

**Destrucción o desmontaje** solo hay una función que se llama componentWillUnmount

**Redux**

Redux es de uso libre que almacena todo el estado de nuestra aplicación en un solo lugar.

**Principios**:

**Almacenamiento** usarlo en los estados se almacenen en un solo lugar.

**Inmutables** que siempre se creen y nunca se sobrescriban.

**Centralizado** toda nuestra aplicación en un solo lugar.

**¿Como funciona?**

En lugar de tener un estado local, lo maneja en un estado global.

¿Cuándo utilizar Redux?

En aplicaciones grandes, en donde el flujo de la información sea compartido por muchos componentes.

Cuando quieres compartir información, por ejemplo, en un **login.**

Es para manejar información.

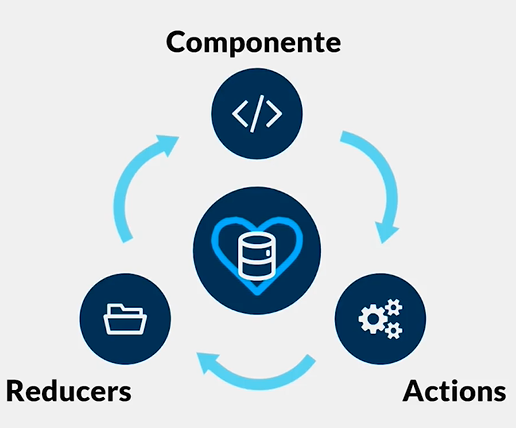
**4Pilares de Redux**

*Store* -> Almacenamiento -> va a hacer el principal porque ahí se va a almacenar todo lo que necesitamos.

*Reducers* -> Estado -> dependiendo del componente la información que vamos a tener.

*Action Creators* -> Funciones -> Son todas las funciones que se llaman para pedir información que generalmente son promesas.

*Componente* -> Código JSX -> Son las partes de React.



**Conexión a un componente**

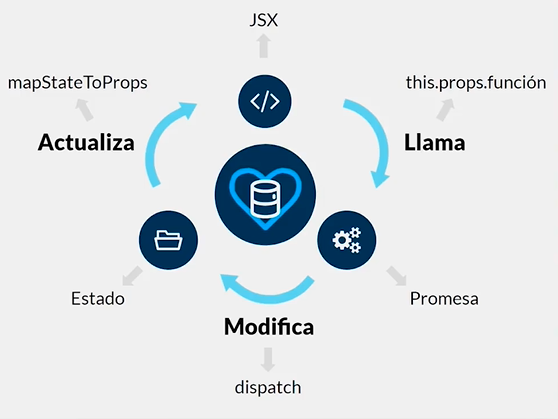
Connect (todos los Reducers que el proveedor le va a entregar al usuario) ()

**Explicación teórica: ciclo completo de Redux**

El componente (JSX) ya carga la interfaz y cuando termina en el componentDidMount (), va y llama al Action creator.

Action creator tiene la promesa porque se esta comunicando con la URL, trae los datos del usuario y luego se los entrega al reducer, que en este caso va y modifica al reducer, lo modifica para que el reducer cree un estado diferente y cree este estado nuevo con lo que se modifica y hace la modificación con el dispatch (tiene dos valores, type y paeolad).

Cuando llega al reducer, el reducer viene siendo el estado, cunado se modifica vuelve a crear un estado. Y ese estado nuevo va y actualiza el JSX. Estos se comunican por el mapStateToProps ().



**Conocimientos adquiridos**

Hemos utilizado React, estado local y luego pasando nuestras propiedades o estados por herencia.(de un padre pasándole a un hijo)

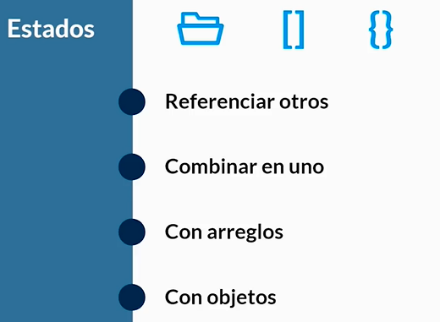


Utilizamos Redux que es en donde estamos, estamos usando React y aparte usando Redux como uno solo lo que significa que nuestro estado lo tenemos de manera global.

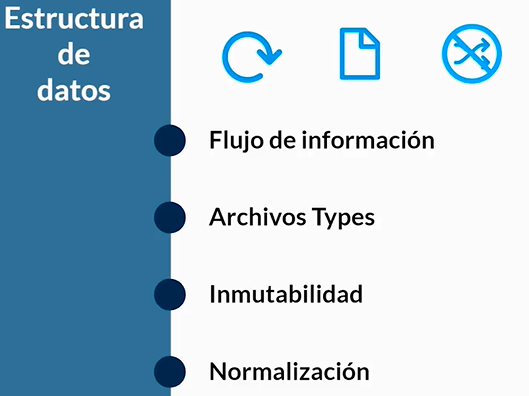
Como lo vimos podemos tener un componente que use solo su estado o que use dos estados o Reducers.

También podemos compartir información entre los Reducers.

Lo que aprendimos:

Estados-> referenciar a otros estados, combinarlos en uno mismo y utilizarlos en arreglos u objetos.

Estructura de datos-> como Redux funciona, nos hicimos autoayuda para poder identificar algunos errores por ejemplo el archivo Types.

La inmutabilidad siempre los estados deben ser inmutables.

Asincronía-> tres estados de la llamada obligatorios *cargando, exitoso y error* toda llamada asíncrona debe de pasar o tener estos estados.

También vimos todos los métodos http que se usan con más frecuencia Post, Get, Put y Delete.

Aprendimos a redireccionar con la etiqueta Redirect con esta nosotros podíamos ver que si se cumplía una condición nos redireccione a la otra página.

Finalmente, las validaciones usamos trycatch para saber cuándo hacer la fase del exitoso y cuando hacer la fase del error. (para decirle al usuario)

