React - Redux

Tags	Front End · Javascript · React · Redux
Última actualización	@May 21, 2021

¿Qué es Redux?

Redux es una librería que nos permite guardar el estado de aplicaciones de Javascript. Es decir, la ventaja que nos da Redux sobre utilizar el estado normal de React es que sacamos algunos estados de los componentes, y los extraemos a un **estado global**, al mover estos valores a este estado global podemos reutilizar los valores con mayor facilidad, y además modificar los valores contenidos en este estado global con mayor facilidad también a lo largo de toda la aplicación.

Anatomía de Redux

Redux consta normalmente de 3 partes: el `store` que es nuestro **estado global**, los `reducers` que son funciones que **manejan casos** y modifican el `store` dependiendo del caso que reciban, y las `actions` que son justamente los casos que reciben los `reducers`.

En redux, los componentes consumen partes del estado global o `store`, o valores directos, y redux les notifica de cambios en ellos para volver a renderizar estos componentes, así como cuando utilizamos el estado propio de los componentes con `setState` o `useState` estas funciones también notifican a React que hubo cambios.

Los componentes pueden **despachar** acciones de redux, al despachar estas acciones, creamos un objeto que tiene un **tipo de acción** (`type` en inglés) y una **carga** o valor (`payload` en inglés).

Estas acciones que despachamos por medio del `dispatch` son enviadas al `store`, este `store` puede estar compuesto de uno o más reductores o `reducers`, cada caso debe ser manejado por un único `reducer`, pero cada reducer puede manejar varios casos. El store decide automáticamente cuál reducer es el que debe llamar dependiendo del caso o type de la acción que le enviamos.

El reductor, una vez que recibe la carga, devuelve una ***nueva copia del estado***, esto porque en redux el estado global es ***inmutable***, es decir, no lo podemos nada más volver a asignar o volver a asignar alguna de sus partes, sino que al devolver un nuevo ***estado completo***, le caemos encima por decirlo de alguna forma al que teníamos antes, y al hacer esto, redux notifica a los componentes que estén escuchando que hubo un cambio en el estado y si estaban usando algunos de los valores que cambió, estos componentes se vuelven a **renderizar** para mostrar los cambios.

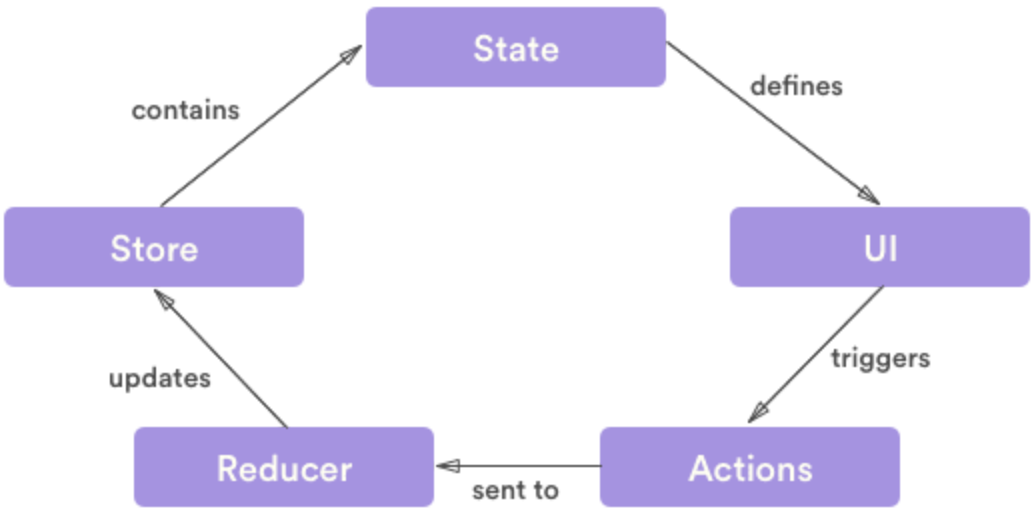

Diagrama de flujo de una aplicación con Redux.

Configuración de Redux

Este paso de realiza una única vez en el proyecto.

El primer paso para agregar Redux a un proyecto de React es instalar los paquetes necesarios:

```
# Si usamos npm:
npm install react-redux redux-thunk

# Si usamos Yarn:
yarn add react-redux redux-thunk
```

¿Qué hace cada paquete?

Redux es el paquete que implementa el store como tal.

1. Creación del primer *reducer*

Para crear nuestro reducer, dentro de la carpeta `src` vamos a crear una carpeta nueva que se llame `store`, dentro de store vamos a crear una nueva carpeta que se llame `reducers`.

En `/src/store/reducers` vamos a crear un archivo con el nombre que le vamos a poner al reductor, dentro de este archivo vamos a escribir el siguiente código.

```
const initialState = {
  nombre: null
}

export default function (state = initialState, {type, payload}) {
  // Un reducer siempre debe ser un switch, para poder agregarle casos.
  switch (type) {
    case 'ACCION_UNO':
      return {...state, nombre: payload}; // Devolvemos una copia nueva del estado

    default:
      return state;
  }
}
```

2. Creación del primer *action*

Para crear nuestra primera acción vamos a crear una nueva carpeta dentro de `store`. Esta carpeta se va a llamar `actions`, dentro de esta carpeta vamos a crear un archivo llamado `actionTypes`, en este archivo vamos a guardar todas las strings que van a representar las los `types` o tipos de acciones que vamos a ir creando. El contenido de este archivo por ahora va a ser:

```
export const ACCION_UNO = 'ACCION UNO';
```

Y ahora vamos a crear un archivo para guardar las acciones dentro del reductor `ejemplo` . En este archivo vamos a definir acciones de redux, las acciones de redux **solo pueden ser objetos planos.** Por defecto las acciones de redux no pueden ser otra cosa que eso, objetos planos**.**

```
import { ACCION_UNO } from './actionTypes';

export const accionUno = (payload) => ({
  type: ACCION_UNO,
  payload
});
```

3. Creando el proveedor

Para crear el proveedor vamos a ir al `App.js` de nuestro proyecto y vamos a importar nuestro reducer, y algunos paquetes de `react-redux` y `redux`.

```
import React from 'react';

// Redux genérico, de una vez
import {applyMiddleware, combineReducers, compose, createStore} from 'redux';
import {Provider} from 'react-redux';
import * as reducers from './src/store/reducers';
import thunk from 'redux-thunk';

const store = createStore(
  combineReducers(reducers),
```

```
  compose(applyMiddleware(thunk)),
);

const App = () => {
  return (
    <Provider store={store}>
      <SafeAreaView>
        {/* componentes aqui */}
      </SafeAreaView>
    </Provider>
  );
};

export default App;
```