MEDAC
Instituto Oficial de Formación Profesional

Redux, iniciación

...reducer, action, store,...

Sergio José Ruiz Cañas

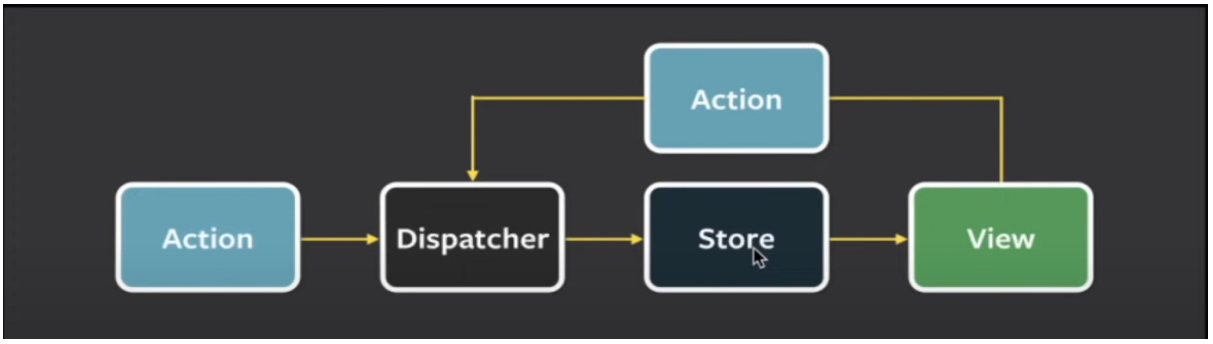
Enlaces de interés

REDUX DEV TOOLS
REACT REDUX

¿Qué es REDUX?

Es una biblioteca, que te puedes intalar, pero tambien es un patrón, es decir, un conjunto de patrones de cómo debes tratar el estado de tu aplicación, es interesante ya que al ser un patrón puedes hacer un redax, sin redax.

Este patrón tiene unas cuantas palabras claves que tienes que dominar, siendo:



Vamos a empezar, crearemos una aplicación que sea un contador y usaremos redux para manejar la lógica de la actualización del estado de nuestro controlador.

Los reducer: concepto base, parecido al reducer() de js, véase:

const reducer = (state, action) => newState

A nuestro reducer le llega un estado y una acción, así, este tiene que expulsar el nuevo estado.

1º Creamos nuestro reducer:

```
const counterReducer = (state =0, action) => {
  switch (action.type) {
    case '@counter/incremented':
      return state + 1;
    case '@counter/decremented':
      return state - 1;
    case '@counter/reseted':
      return 0;
    default:
    return state
  }
}
```

Se suele encontrar así los reducer, en el caso de que la acción sea decremented, tal, incremented return tal...

2ºCreamos nuestras acciones:

```
const actionIncremented = {
  type: '@counter/incremented'
}

const actionDecremented = {
  type: '@counter/decremented'
}

const actionReset = {
  type: '@counter/reseted'
}
```

En las acciones, más de lo mismo, declaramos la acción y en ella almacenamos el tipo. tipo contador/acción.

3ºNuestra Store:

Objeto que reune las acciones y reducers. Sus responsabilidades son:

-Contener estado de la aplicación.

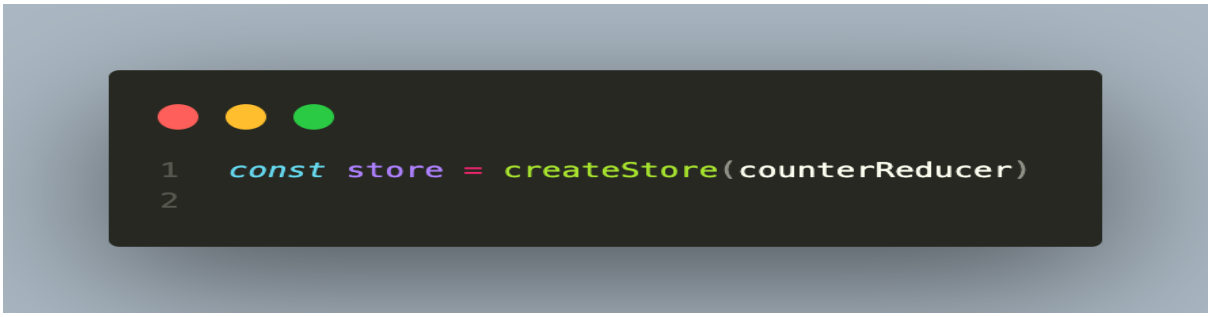
-Permite leer el estado de la aplicación.

-Actualizar el estado de la aplicación.

Para ello: import {createStore} from 'redux'

**Recuerda: npm install --save redux react-redux

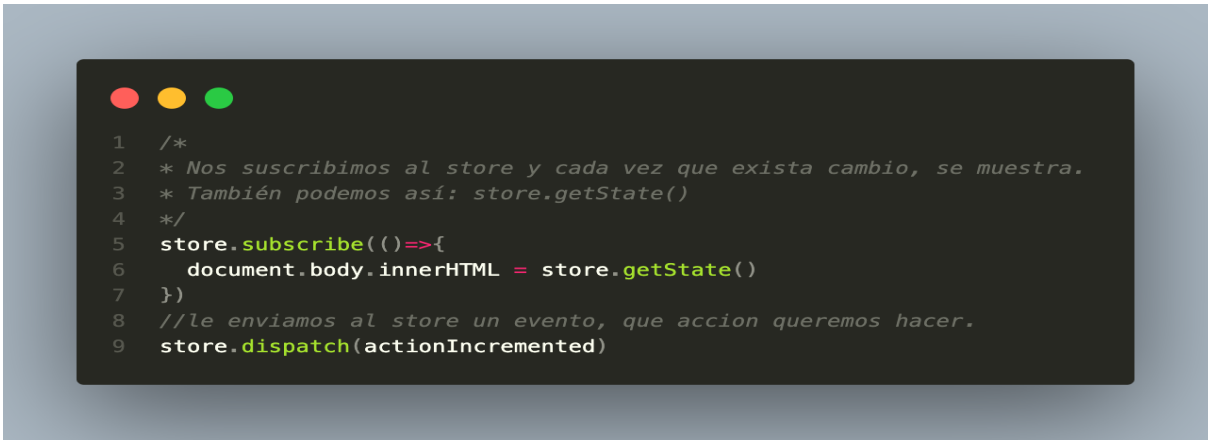
Para usar la store, le pasamos el reducer que tiene que usar la store:



```
const store = createStore(counterReducer)
```

Así sabe, al pasarle el counterReducer, como tiene que actualizar el estado.

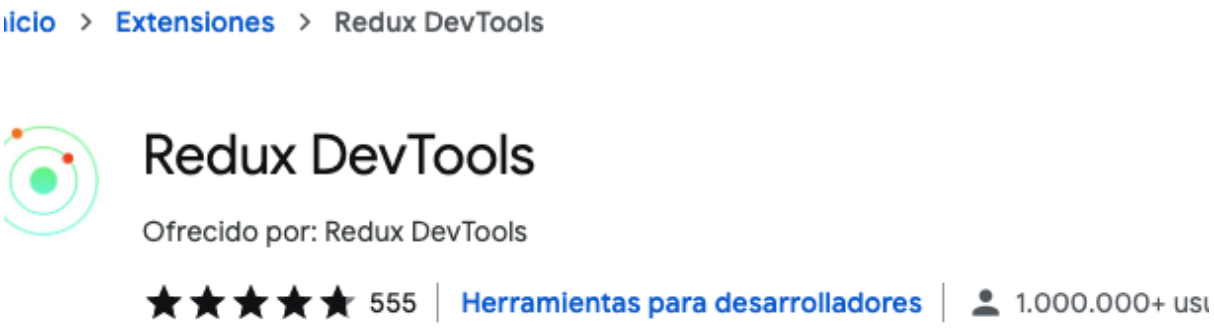
Ejemplo sencillo de dispatch:



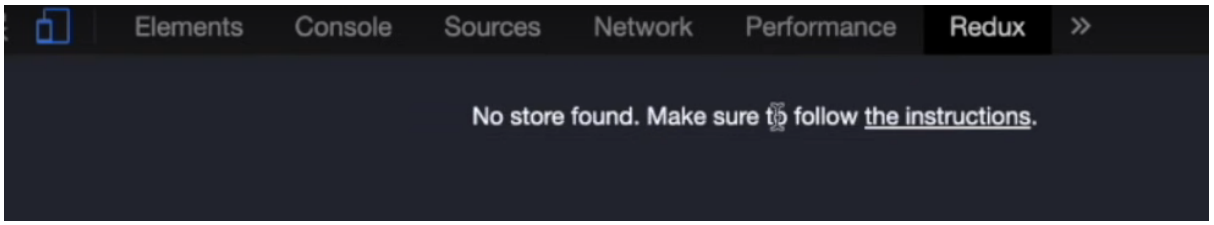
```
/*
* Nos suscribimos al store y cada vez que exista cambio, se muestra.
* También podemos así: store.getState()
*/
store.subscribe(()=>{
  document.body.innerHTML = store.getState()
})
//le enviamos al store un evento, que accion queremos hacer.
store.dispatch(actionIncremented)
```

RECOMENDACION: REDUX DEV TOOLS

Nos ayuda a entender mejor nuestra aplicación, a resolver problemas con esta,



Para utilizarla, se debe añadir ya que si no nos aparece esto:



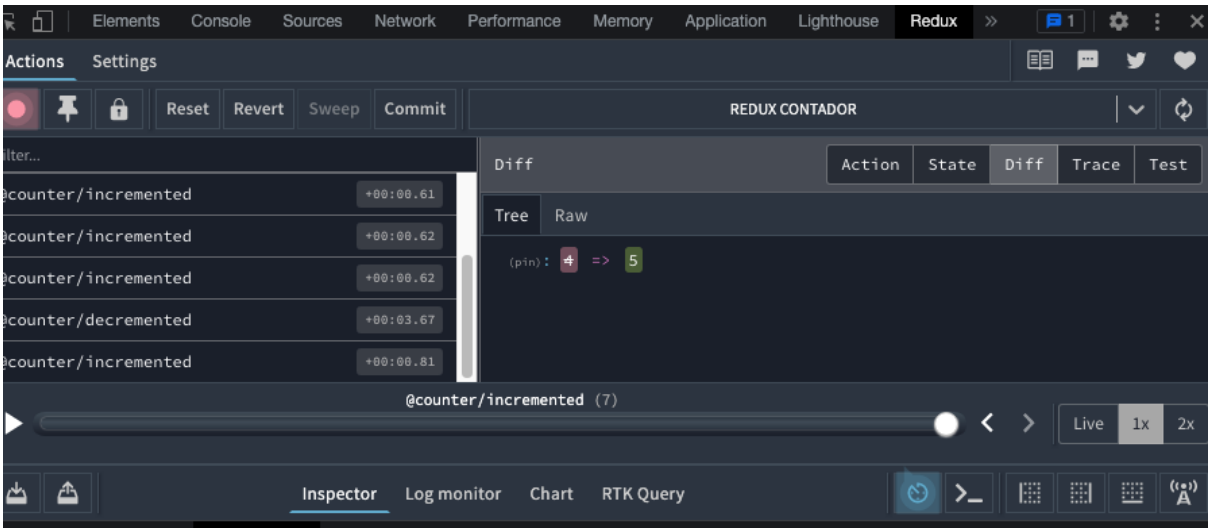

```
const store = createStore(
  reducer, /* preloadedState, */
 window.__REDUX_DEVTOOLS_EXTENSION__ &&
window.__REDUX_DEVTOOLS_EXTENSION__()
 );
```

**Recalcar que si haces uso de librerías más grandes, puede ser que ya venga incluido, no es este caso. Y, que esto solo se puede ejecutar en el cliente, en el caso de hacerlo en el lado servidor, crashea.

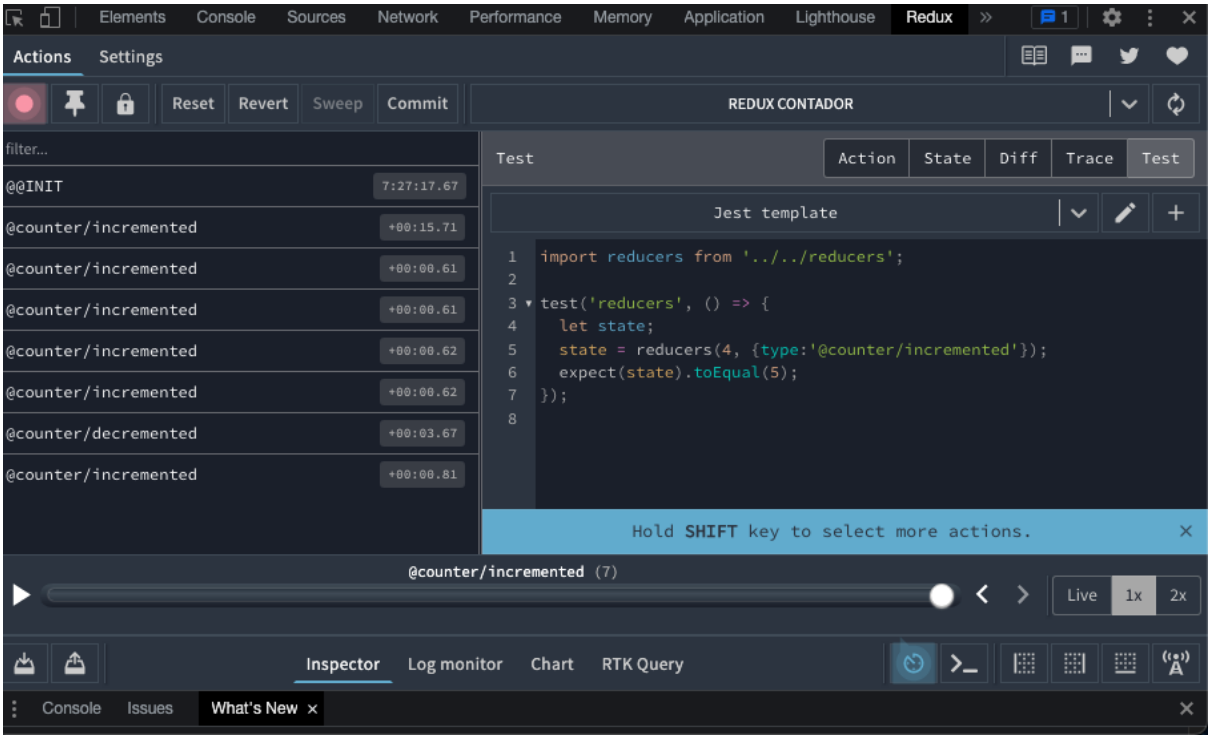
en el proyecto en nuestro createStore():

```
const store = createStore(
  counterReducer,
  window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__()
)
```

aspecto en consola:



También, nos ofrece tests, que solo tenemos que copiarlos y darle uso:


Elements
Console
Sources
Network
Performance
Memory
Application
Lighthouse
Redux
Actions
Settings
Reset
Revert
Sweep
Commit
REDUX CONTADOR
filter...
@@INIT 7:27:17.67
@counter/incremented +00:15.71
@counter/incremented +00:00.61
@counter/incremented +00:00.61
@counter/incremented +00:00.62
@counter/incremented +00:00.62
@counter/decremented +00:03.67
@counter/incremented +00:00.81
Test
Action
State
Diff
Trace
Test
Jest template
import reducers from '../../reducers';
test('reducers', () => {
let state;
state = reducers(4, {type:'@counter/incremented'});
expect(state).toEqual(5);
});
Hold SHIFT key to select more actions.
@counter/incremented (7)
Live
1x
2x
Inspector
Log monitor
Chart
RTK Query
Console
Issues
What's New