educación IT
CODE YOUR FUTURE

ReactJS Developer

Módulo 8

educación IT
CODE YOUR FUTURE

Redux Middlewares

Redux Middlewares

Los *middlewares* en Redux siguen la misma filosofía que los *middlewares* en cualquier otro tipo de sistema. Simbolizan **una forma de interceptar al código, logrando extenderlo y agregarle funcionalidad entre el momento en que algo se recibe y algo de resuelve.**

Específicamente hablando de Redux, un *middleware* sirve para **realizar o ejecutar un código en el momento intermedio entre que una acción es despachada y la misma llega al reducer.**

Uno de los puntos positivos que tiene trabajar con *middlewares* es que funcionan **en forma de cadena, uno tras otro, realizando cada uno su acción pertinente.**

Casos de uso

Supongamos, por ejemplo, que queremos loguear todas las acciones que llegan al *store* para poder evaluar en desarrollo que está sucediendo. En este caso, una primera aproximación podría ser poner **`console.log`** en todas las acciones que realizamos, lo que no sería para nada bueno.

Pensemos por un segundo que necesitamos, para poder controlar animaciones, que una acción que llega al *store* se retrase **N milisegundos**, que no se despache inmediatamente. Tendríamos que poner un **`setTimeout`** en el `action creator` (que no es lo deseable).

Imaginemos que necesitamos que una acción resuelva de forma **asincrónica**, como sería el caso de que necesitemos pedir información a una API externa.

Todos estos ejemplos y muchos más, son casos en los cuales necesitamos de alguna forma capturar esas acciones y actuar de formas distintas para poder lograr el objetivo.

Para estos casos existen **los *middlewares*.**

Implementación

Al momento de crear un *store*, podemos utilizar una función que nos provee Redux que es **`applyMiddlewares`**.

Es una función que recibe como parámetro **un listado de *middlewares*** y la cual se utiliza al momento de crear el *store*. Primero importamos la función:

```
1  import {createStore, applyMiddleware} from 'redux';
2
3  const store = createStore(todos, ['Use Redux'], applyMiddleware(miMiddleware));
4
```

¿Pero qué sería, en este caso, **`miMiddleware`**?

Un *middleware* es básicamente **una función con la siguiente forma:**

```
const miMiddleware = store => next => action => {
    // store: acceso a todo el store
    // next: acceso a la acción siguiente
    // action: acceso a la acción previa
}
```

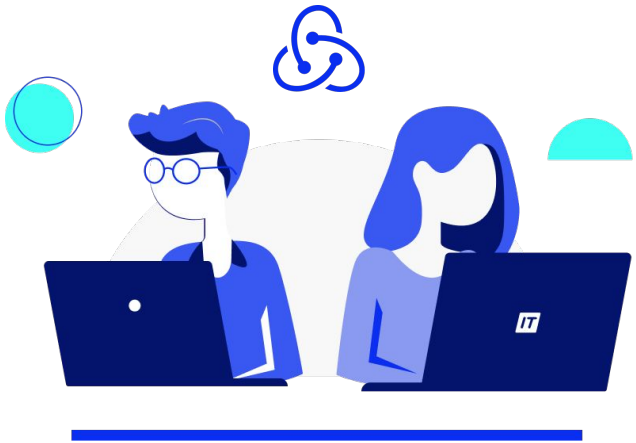
Es decir, es una función que **recibe el `store`** y que devuelve **una función que recibe `next` y devuelve una función que recibe la acción**. Suena como un trabalengua.

Veámoslo en detalle:

- "Es una función que recibe el `store`": hasta acá nada que no podamos comprender.
- "Devuelve una función que recibe `next`": la primera devolución tiene que ser **una función que recibe un parámetro llamado `next`**. **`Next`** es una función que simboliza de forma genérica **el siguiente paso en la cadena de *middlewares***.

  Lógicamente, **si solo tenemos un *middleware* entonces `next` despachará definitivamente la acción al reducer**.
- "Devuelve una función que recibe la acción": es la función final de vuelta que **recibe la acción con la cual vamos a poder trabajar**.

En el código a ejecutar podemos hacer lo que nos sea conveniente y en última instancia determinar **parar la ejecución de la acción, definir qué hacer si se cancela o decidir que siga la cadena invocando a `next(action)`**.

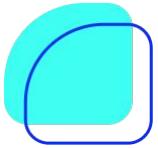


Redux-Thunk

Redux-Thunk

Redux-Thunk es un *middleware* de Redux que sirve para poder **despachar funciones en vez de acciones planas en JSON**.

Esto nos va a dar la posibilidad de manejar a nuestro antojo y con facilidad **los efectos secundarios que necesitemos en nuestra aplicación**, como retrasar una acción o realizar llamadas asíncronas que deban esperar a que finalicen para despachar realmente la acción.



Instalación y uso

Primero que nada necesitamos **incluirlo en nuestro proyecto** con el siguiente comando:

```
npm install redux-thunk
```

Una vez instalado, solo necesitamos **importarlo** y **asignarlo como un *middleware* de Redux**:

```
1   import {createStore, applyMiddleware} from 'redux';
2   import reduxThunk from 'redux-thunk';
3   import rootReducer from './rootReducer';
4
5   const store = createStore(
6       rootReducer,
7       applyMiddleware(reduxThunk) // Incluímos reduxThunk acá
8   );
9
10  export default store;
```

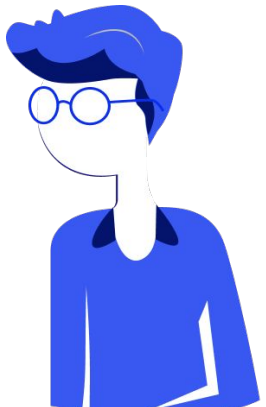
Firma de las funciones de *Redux-Thunk*

El uso de este *middleware* es muy simple: el *middleware* detecta **el tipo de dato de la acción despachada**. Si es una función, entonces la ejecuta directamente pasándole el método **`dispatch`** para que, a su vez, se puedan ejecutar acciones en esa función.

Las funciones que se usen con **`thunk`** tienen que tener la siguiente estructura:

```
const thunkFunction = () => (dispatch) 0> {}
```

En vez de usar `dispatch` con acciones, **usarás `dispatch` con funciones** ¡Eso es todo!



educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!