Projecte Redux JS

M06 UF2

Curs 2021-22

Raúl Heredia i Ana Menéndez



Índex

[Què és Redux? 3](#_Toc94972004)

[Avantatges del store de Redux 4](#_Toc94972005)

[Com funciona Redux? 4](#_Toc94972006)

[Accions en Redux 4](#_Toc94972007)

[Reducers en Redux 5](#_Toc94972008)

[Store en Redux 5](#_Toc94972009)

[Exemple bàsic de funcionament. 7](#_Toc94972010)

[El nostre projecte 8](#_Toc94972011)

[APIS Utilitzades 8](#_Toc94972012)

[Lògica de la aplicació 8](#_Toc94972013)

[Emmagatzematge de dades 8](#_Toc94972014)

[HTML5 Apis 8](#_Toc94972015)

[Web Storage 8](#_Toc94972016)

[IndexedDB 8](#_Toc94972017)

[Drag And Drop 9](#_Toc94972018)

[FileReader 9](#_Toc94972019)

[POO amb JS 9](#_Toc94972020)

[Funcions amb JS 9](#_Toc94972021)

[Arrays amb JS 9](#_Toc94972022)

[Métodes d’array utilitzats 9](#_Toc94972023)

# Què és Redux?

Redux és una llibreria JavaScript de codi obert per al maneig de l’estat de les aplicacions. És comunament utilitzada amb altres llibreries com React o Angular per a la construcció d’interfícies d’usuari.

Al 2015 per Dan Abramov i Andrew Clark es van inspirar en una altra llibreria de Facebook, Flux per crear Redux. Abramov es trobava impressionat per la similitud del patró Flux amb la funció de reduir.

*"Estava pensant sobre Flux com una operació reduir... el teu magatzem, com acumulen un estat en resposta a unes accions. Vaig pensar a anar més enllà. Què si el teu magatzem Flux no fos un magatzem sinó una funció reduir?"*

Abramov va contactar amb Clark per col·laborar junts. Gràcies a Clark hi ha les eines que fan possible l'ecosistema Redux, va ajudar a definir un API coherent i implementar la possibilitat d’extensió mitjançant middleware i store enhancers.

Redux és una petita llibreria amb una API simple i limitada que está dissenyada per ser un contenidor predictible de l’estat de l’aplicació.

El propòsit de Redux és fer predictibles els canvis d'estat, imposant certes restriccions sobre com i quan es poden produir les actualitzacions. Redux aconsegueix que la teva gestió d'estat sigui transparent i determinista, cosa que entre altres coses aporta:

* Millor comprensió de l’evolució de l’estat en un moment donat.
* Facilitat per incorporar noves característiques a l’app.
* Nou ventall d’eines de debugging.
* Capacitat de reproduir un bug.
* Millores en el procés de desenvolupament i poden reiniciar l’execució a partir d’un estat concret.

## Avantatges del store de Redux

* Model de dades més consistent i segur
* Gestió senzilla de múltiples fonts de dades:
* Dades accessibles en temps real:

# Com funciona Redux?

La manera com funciona Redux és senzilla. Hi ha una “Store” central que conté tot l'estat de l'aplicació. Cada component pot accedir a l'estat emmagatzemat sense haver d'enviar accessoris d'un component a un altre, això resulta molt util en aplicacions de Angular o React que tenen diferents components.

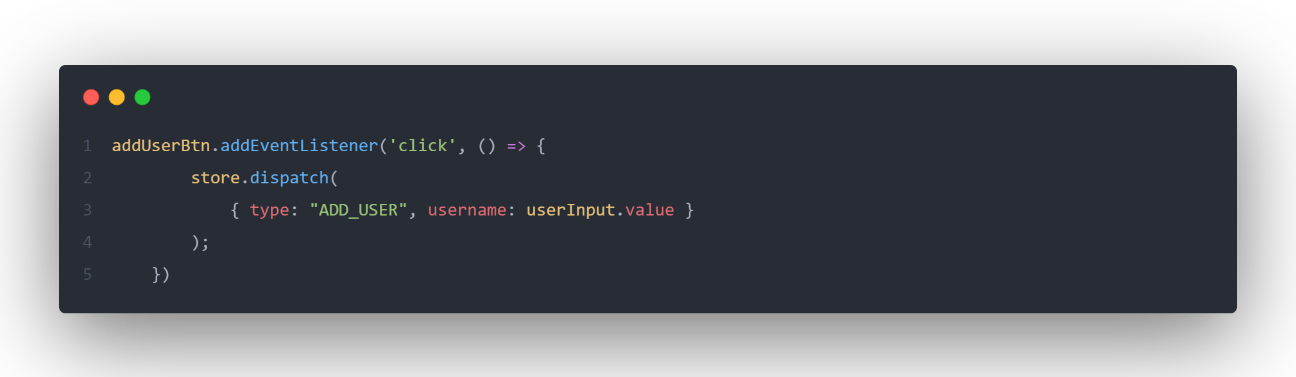
Principalment, Redux es divideix en tres parts: **accions**, **Store** i **reductors**. Ara, veurem que fan cadascuna.

## Accions en Redux

En poques paraules, les accions són esdeveniments. Són l'única manera d'enviar dades de la teva sol·licitud a la nostra Store de Redux. Les dades poden provenir d'interaccions d'usuari, trucades d'API o fins i tot d'enviaments de formularis.

Les accions s'envien mitjançant el mètode store.dispatch(). Les accions són objectes JavaScript simples, i han de tenir una propietat tipus per indicar el tipus d'acció a dur a terme. També han de tenir una càrrega útil que contingui la informació que ha de ser treballada per l'acció. Les accions es creen a través d'un creador d'accions.

**Per exemple:**



En aquest cas, quan cliquem el botó afegir usuari, enviem una acció de tipus “ADD\_USER” amb la propietat “username” i el valor agafat d’un input.

## Reducers en Redux

Els reductors són funcions pures que prenen l'estat actual d'una aplicació, realitzen una acció i retornen un nou estat. Aquests estats s'emmagatzemen com a objectes i especifiquen com canvia l'estat d'una aplicació en resposta a una acció enviada a la store.

Es basa en la funció de reducció en JavaScript, on un sol valor es calcula a partir de múltiples valors després de realitzar una funció de resposta.

**Per exemple:**



En aquest cas, veiem que el reducer està composat per l’estat i la acció, cada vegada que fem un dispatch, la acció arriba al reducer. Com podem veure, si la petició es de tipus “ADD\_USER”, retornarà l’estat més el nom d’usuari que li acabem de passar, en cas de que la acció no sigui de tipus “ADD\_USER”, simplement retornarà l’estat sense fer canvis

Store en Redux

La Store, té l'estat de l'aplicació. És molt recomanable mantenir només una Store en qualsevol aplicació Redux. Amb la Store, podem accedir a l'estat emmagatzemat, actualitzar l'estat i registrar o deixar de registrar listeners.

Els mètodes més utilitzats son Store.subscribe() i Store.getState(). Subscribe es un mètode que es truca cada vegada que canvia l’estat de la nostra aplicació, getState serveix per obtenir l’estat de la nostra aplicació.

**Per exemple:**



Com podem veure, fem un subscribe amb una funció anònima, això es el que farà cada vegada que s’actualitzi l’estat. En aquest cas el que fem es eliminar el contingut d’una llista HTML i el value del input d’usuari, seguidament fem un getState i iterem mitjançant un forEach, per cada usuari que hi hagi a l’estat crearem un element ‘li’, el qual tindra una classe de bootstrap i el text que contindrà serà l’usuari, finalment fem un appendChild del element ‘li’ per afegir-lo a la llista.

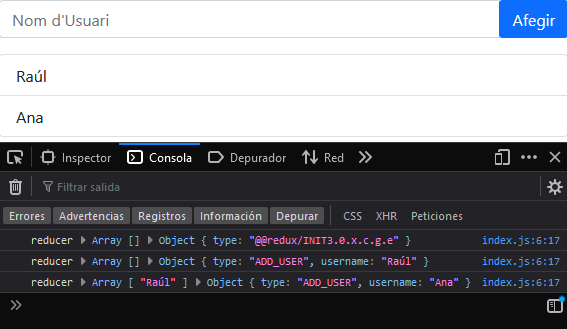
## Exemple bàsic de funcionament.

Texto

Descripción generada automáticamente

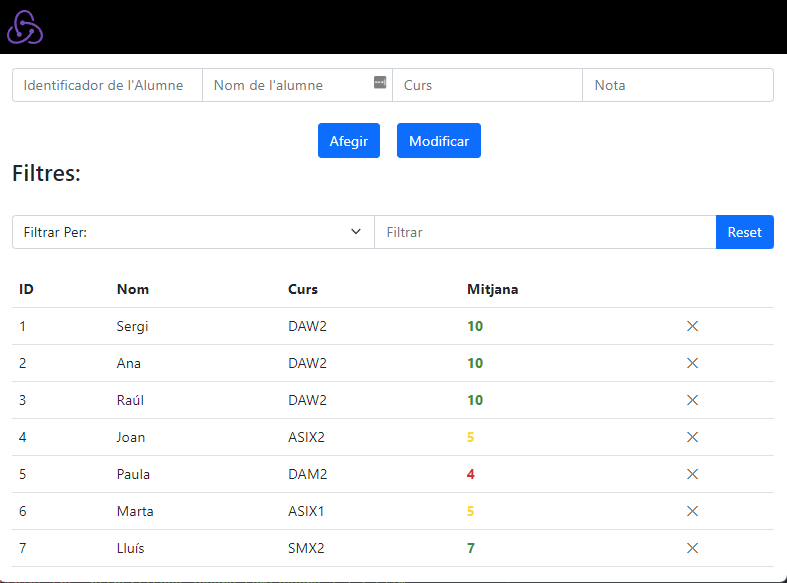
En aquest cas, podem veure un codi sencer, amb el reducer, la Store i la acció que em vist anteriorment.

Cada vegada que introduim un usuari i fem clic al botó afegir, s’envia una acció al reducer mitjançant un dispatch i canvia l’estat, llavors amb el subscribe cada vegada que aquest canvia, actualitza la llista:



# El nostre projecte

El projecte es una aplicació web que permet afegir alumnes, amb un identificador, nom curs i nota. En cas de voler modificar-ne algun, simplement posem el identificador, la dada que volem modificar i fem clic al botó de modificar. Podem filtrar els alumnes per aprovats i suspesos, a més de poder buscar-los per nom o curs, escrivint al input anomenat filtrar.



## APIS Utilitzades

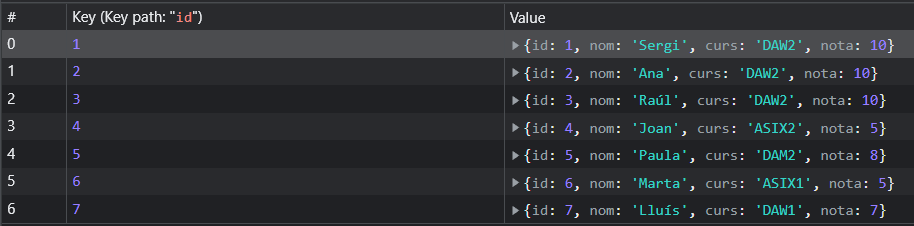
### [Web Storage](https://developer.mozilla.org/es/docs/Web/API/Web_Storage_API)

Hem utilitzat Web Storage per emmagazemar únicament el ID de l’alumne i el seu nom. Quan carrega la pàgina obté les dades de IndexedDB, cada vegada que afegim un usuari, es guarda el ID i el nom, quan eliminem un usuari, s’elimina també, es guarda mitjançant Local storage i no Session Storage. A més la utilitzem per a mostrar les dades per consola mitjançant una funció dinàmica que es veurà més endavant.



### [IndexedDB](https://developer.mozilla.org/es/docs/Web/API/IndexedDB_API)

Hem utilitzat IndexedDB per a guardar els objectes dels alumnes. La key que utilitza es el id d’usuari. Al entrar a la aplicació, IndexedDB envia cada usuari al reducer de la api Redux i d’aquesta forma els carreguem en el estat, que es mostrat en la taula d’alumnes



### [Drag And Drop](https://developer.mozilla.org/es/docs/Web/API/HTML_Drag_and_Drop_API/File_drag_and_drop)

Hem utilitzat la API de drag and drop per a poder arrossegar arxius JSON, amb els quals podem afegir alumnes de forma massiva.

### [FileReader](https://developer.mozilla.org/es/docs/Web/API/FileReader)

Hem utilitzat la API FileReader per a processar les dades del fitxer.

## Lògica de la aplicació

## Emmagatzematge de dades

## POO amb JS

Hem creat una classe emmagatzematge que conté dos mètodes estàtics. El mètode desar

Texto

Descripción generada automáticamente

També, hem creat una classe anomenada WebStorageEmmagatzematge que hereta de Emmagatzematge, aquesta compta amb un override dels dos mètodes de la superclasse.

Texto

Descripción generada automáticamente

## Funcions amb JS

## Arrays amb JS

### Métodes d’array utilitzats

map(),

filter()

sort()

splice()

forEach()

reduce() (Relatiu ja que es la funció reducer que uitlitza redux

for in