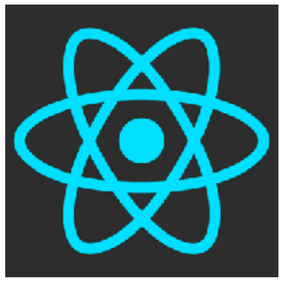
Accedo

React Training

Episode 2 - return of the props

Greg Desfour

React

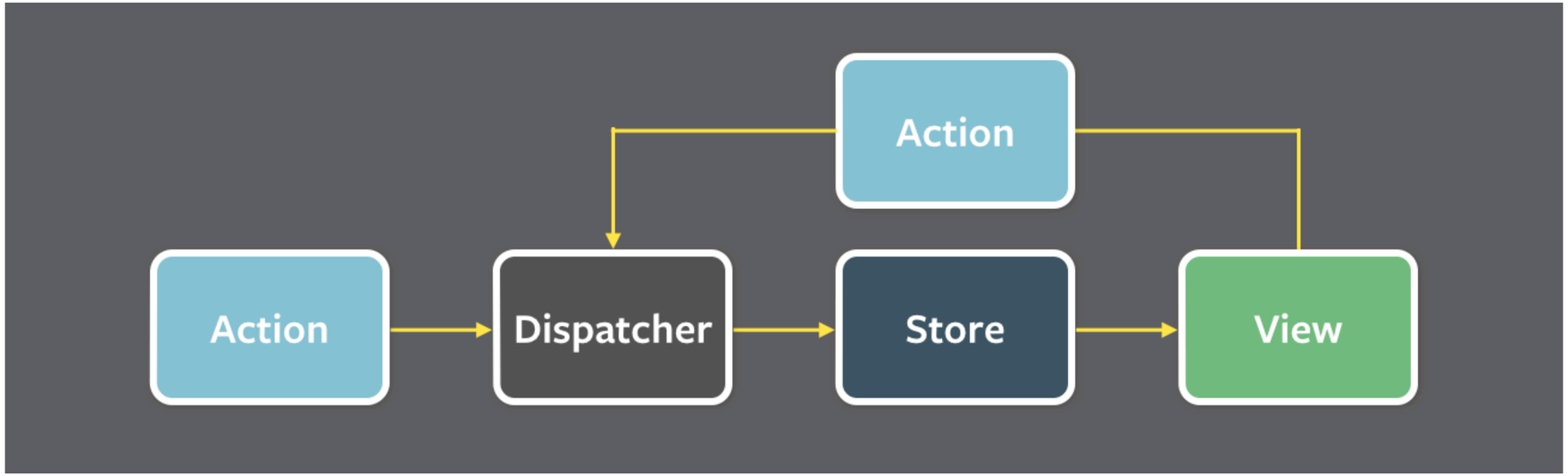


- It’s basically Components + VDOM reconciliation
- That’s OK for a ‘todo list’ or building a counter…
- …But how to structure larger apps ?

Challenges

- Communicating between components
- Communicating with the outside world (side effects)
  - XHR / Fetch
  - Browser Storage / IndexedDB …
- Props, state, callbacks help. Keeping it organised is not so easy.

∞ Flux



- Initial suggested way to structure React apps, rather than MVC
- Unidirectional data flow

Redux

- Takes ideas from Flux, Om (a ClojureScript lib), and the ‘Elm Architecture’.
- Only one store
- The store and actions are serialisable (JS objects similar to JSON)
- Adds some overhead / boilerplate, helps maintenance

Redux principle

- Components are given (parts of) the **store**, and **actions** they can dispatch
- Actions are read by **reducers**, which replace parts of the store

myReducer = (oldState, action) => newState

- By the way, this is not only for React ! Angular, Deku, Preact, Inferno and others can use it too.

Actions

- Just a plain JS object
- Most people use the FSA format (Flux Standard Action): each action has a **type** property, and an **error** and/or a **payload**
- install: `yarn add redux-actions`

Reducers

- Functions that, based on the current state and the action that was received, return a new state.
- It should be pure functions (the result depends only on the input, same input leads to same results)

Store

- A simple JS object
- Usually made up of several properties, each handled by a different reducer.
- So a reducer can only ‘modify’ the part of state it’s responsible for.

Let’s try 😎

- Start with the previous simple React component we had last time, with some extra boilerplate added to be ready to work with Redux.
- https://github.com/gouegd/react-training-ep2
- Check how the integration of Redux was started

Connecting React to Redux

- At least one component should be connected to Redux
- It will receive the parts of the state and actions it is interested in
- FYI, this is done through a HOC (remember ? 😱) to access the React context (remember ? 😰) - doc
- After this, your compos can access the redux store. Try it out.
- Sample solution

Remember the devtools !

- You can inspect the Redux store with the Redux devtools
- You can what values were passed into a component with the React devtools.

Actions and action creators

- Action creators are just functions to create FSA-compliant actions
- The redux-actions lib has helpers to create action creators (this is optional)… 🤷‍♂️
- Create some, then you can let your connected component access them, and trigger them.
- See how actions they appear in the redux dev tools.
- Sample solution

Reducers

- Now you can also modify the reducer to transform the state when the proper action is triggered.
- If you do this and your component(s) use the store to render something, things will just work as expected when you trigger the right action. ✌️
- Try it out, and see how some state change appears in the dev tools now.
- Sample solution

Async (side effects) 1/2

- Redux accepts middlewares that can affect its transformation pipeline. With this, it can handle actions of different types.
- Let’s install redux-promise and redux-thunk: they can handle Promise and thunk (function) actions. It’s a popular and simple way to handle side effects.
- See redux-promise install and test sample
- See redux-thunk install and test sample

Async (side effects) 2/2

- Now you could take advantage of those new tools to clean up the component, moving the network call to be a side effect of an action.
- The component is then more reusable, the business logic decoupled from it - it only deals with rendering and triggering actions.
- See sample solution

FYI

- There is another popular way to handle side-effects, it's using Sagas (redux-saga) with ES6 generators.
- It replaces redux-promise and redux-thunk - or you can use both if you like.
- It's harder to grasp, but much simpler to handle complex async flows.

That’s it ! 👋

Bra jobbat ! Trevlig helg !

Accedo