JavaScript - Lesson 3
Enhanced Applications

Intervenant

- Quentin Pré
  <pre.quentin@gmail.com>

- MTI 2014

ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

- Lead Front-End Engineer
  @ Adikteev / MotionLead

- I make digital ads for a living,

- I'm pretty sure there is a
  special place in hell for this.

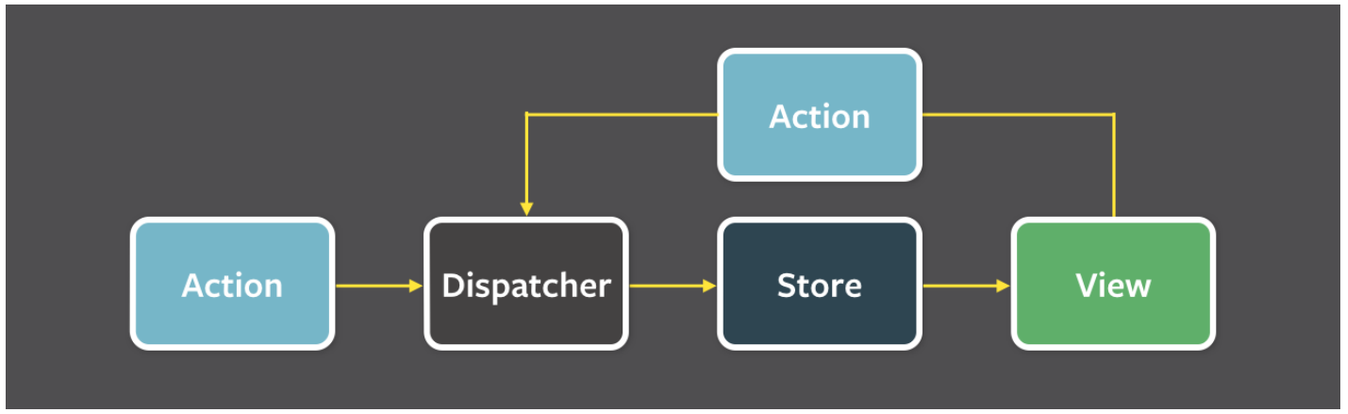
adikteev

motionlead

Redux

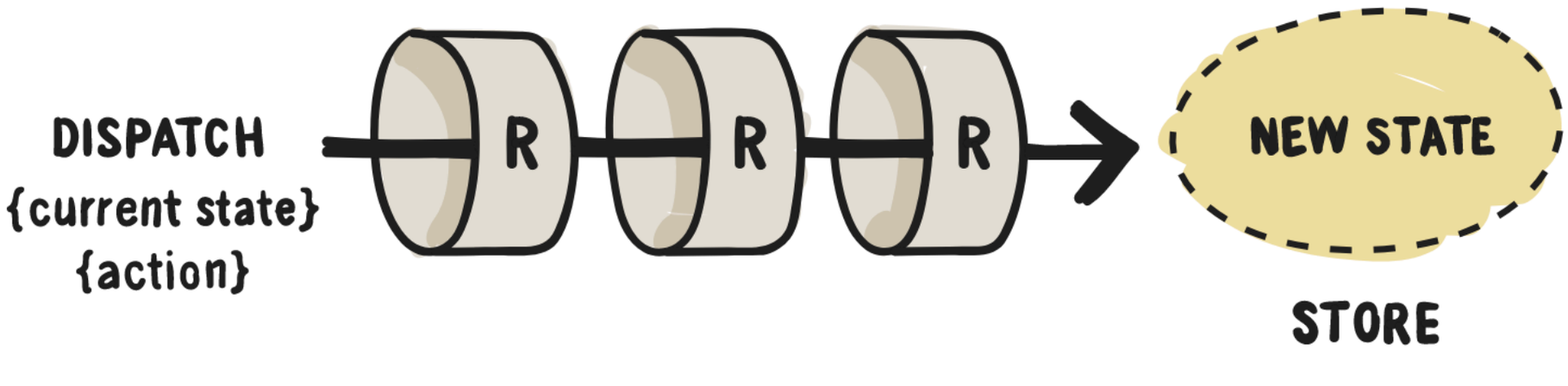
What the Flux ?

- Applications are gaining in complexity
- It’s getting hard to keep track of what, why and how something happens.
- Facebook’s Flux relies on unidirectional data flow



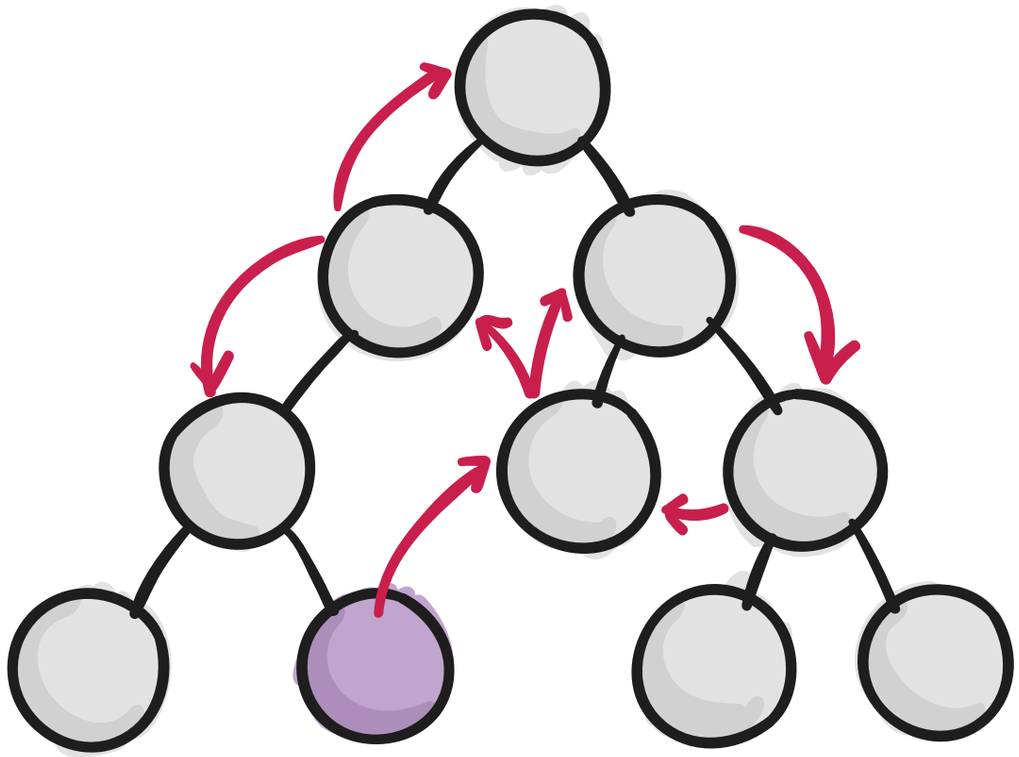
- Redux takes inspiration (even though it’s not a strict implementation of it)

Redux: Three Principles

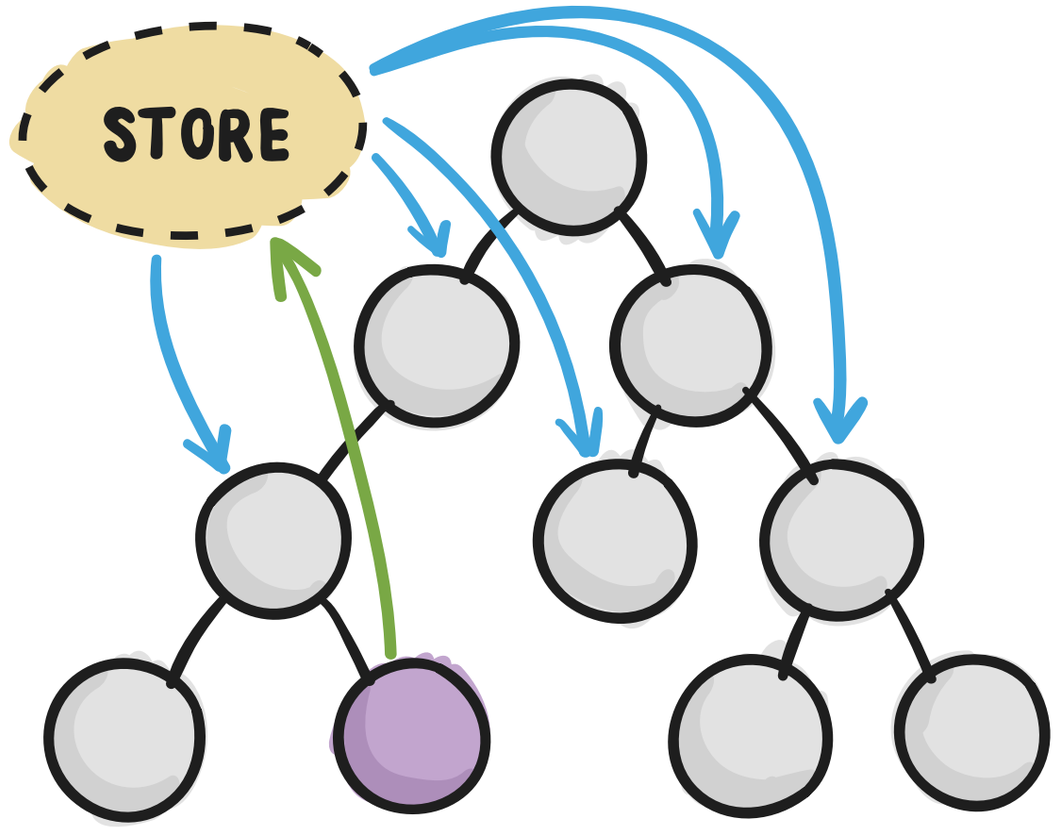


- **Single Source of Truth**: There is a single state, in a single store.
- **State is read-only**: the only way to change state is to dispatch an action with will spawn a new state.
- **Changes are made with pure functions**: aka ‘reducers’

WITHOUT REDUX



WITH REDUX



COMPONENT INITIATING CHANGE

A basic Redux app

```
import { createStore } from 'redux'

function counter(state = 0, action) {
  switch (action.type) {
  case 'INCREMENT':
    return state + 1
  case 'DECREMENT':
    return state - 1
  default:
    return state
  }
}

let store = createStore(counter)

store.subscribe(() => console.log(store.getState()))

store.dispatch({ type: 'INCREMENT' }) // => 1
store.dispatch({ type: 'INCREMENT' }) // => 2
store.dispatch({ type: 'DECREMENT' }) // => 1
```

Reducers

- Reducers are pure functions
- They return a new state.
- combineReducers
- A reducer’s name will be the key to access its state from the store.

```
import { combineReducers, createStore } from 'redux';

const dumb = (state = [], action) => state;
const dumber = (state = [], action) => state;

let rootReducer = combineReducers([
  dummy,
  counter,
]);

let store = createStore(rootReducer);

const { dumb, dumber } = store.getState();
```

Action Creators

- **Actions** are objects with a type field
- **Action Creators** are functions returning actions
- use **dispatch** to propagate your actions.

```
const PICTURE_ADD = "PICTURE_ADD";

function addPicture(data) {
  return {
    type: PICTURE_ADD,
    data,
  };
}

const data = { src: 'https://serv.er/picture/123' };
store.dispatch(addPicture(data));
```

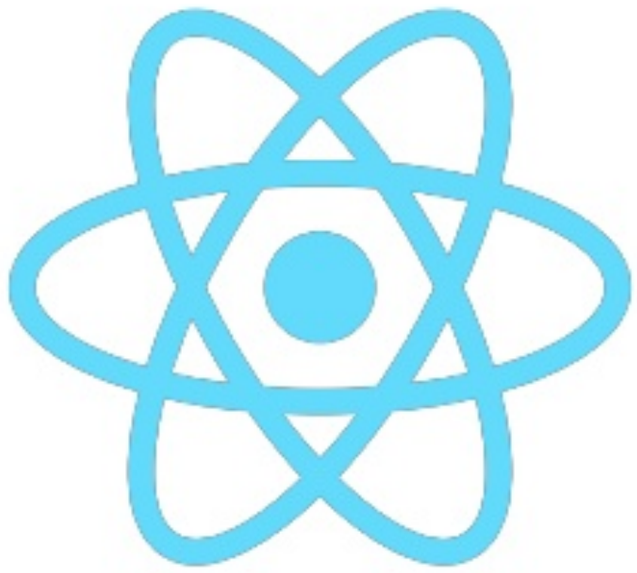
Redux-thunk

- **Redux-thunk** is a middleware for Redux, it allows you to return functions instead of actions in your Action Creators.
- The returned function will be given (dispatch, getState) as arguments.
- Use it to tame asynchrony in your app.

```
const REQUEST_PICTURE = 'REQUEST_PICTURE';
const RECEIVE_PICTURE = 'RECEIVE_PICTURE';

const receivePicture = (data) => ({ type: RECEIVE_PICTURE, data });
const requestPicture = () => ({ type: REQUEST_PICTURE });

funtion fetchPicture(picId) {
  return (dispatch, getState) => {
    dispatch(requestPicture());

    fetch(`https://serv.er/pictures/${picId}`)
      .then(() => dispatch(receivePicture));
  }
}

store.dispatch(fetchPicture(123));
```

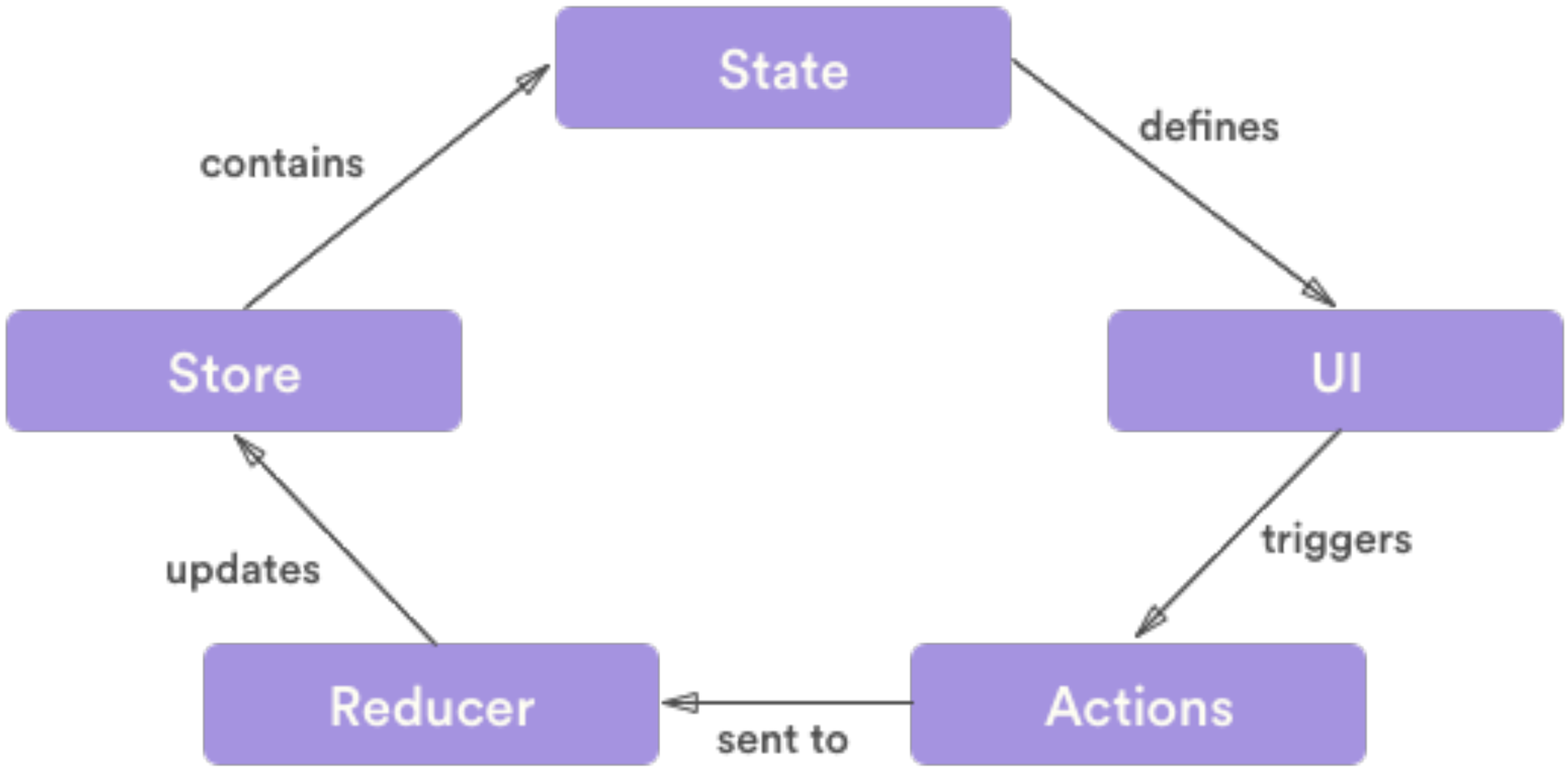
Debug

- https://github.com/zalmoxisus/redux-devtools-extension



React + Redux

react-redux



- NPM package react-redux
- Allows you to connect your React app to your Redux store

<Provider />

- Provider is a React Component

- It provides an access to the **store** to all its children

- Therefore it must be at the top of your tree

```
ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  rootEl
)
```

connect

- **Connect** gives you a **Higher Order Component**, give it your component and get a connected component

- You can use it to cherry pick relevant informations from the App state to pass to your connected component as props

- A common pattern is to have “Controller Components” or (layouts, container…) connected to the state, that pass information to dummy components via props

```
// inject 'dispatch' as a prop
export default connect()(Component)

// inject dispatch and get props from store
function mapStateToProps(state, ownProps) {
  const pic = state.pictures.find(p => p.id === ownProps.id);

  // expose all 'pic' fields as component props
  return { ...pic };
}

export default connect(mapStateToProps)(Picture);

// NEVER do this
export default connect(state => state)(Component);
```

More & Credits

React-Redux has an amazing community (too)

React was initiated by Jordan Walke, who's also working on ReasonML

An EPITA alumni is part of React Native core team @vjeux (he's also been a great influence on React and also works on prettier)

A keystone of the ecosystem is Dan Abramov, co-creator of Redux and now core-member of the React team

Checkout @_chenglou's talk on the cost of abstraction
(I think he was an intern at Facebook at the time…)

Checkout Reason-React

Have a question later ?

<pre.quentin+mti2019@gmail.com>

“Gouter, on va gouter !”

– Richard Bullit

Get the workshop assets here: https://we.tl/1a1cIiRAWH