Utiliser Redux avec Angular

Created by Julien Poyard / @jpoyard

Julien Poyard

@jpoyard

15 années d'expérience

utbm
université de technologie
Belfort-Montbéliard

OXiane

et vous?

Connaissez-vous ...

- ... Javascript ES2016/2017?
- ... TypeScript?
- ... Angular?
- ... ReactiveX?
- ... RxJS?
- ... et Redux?

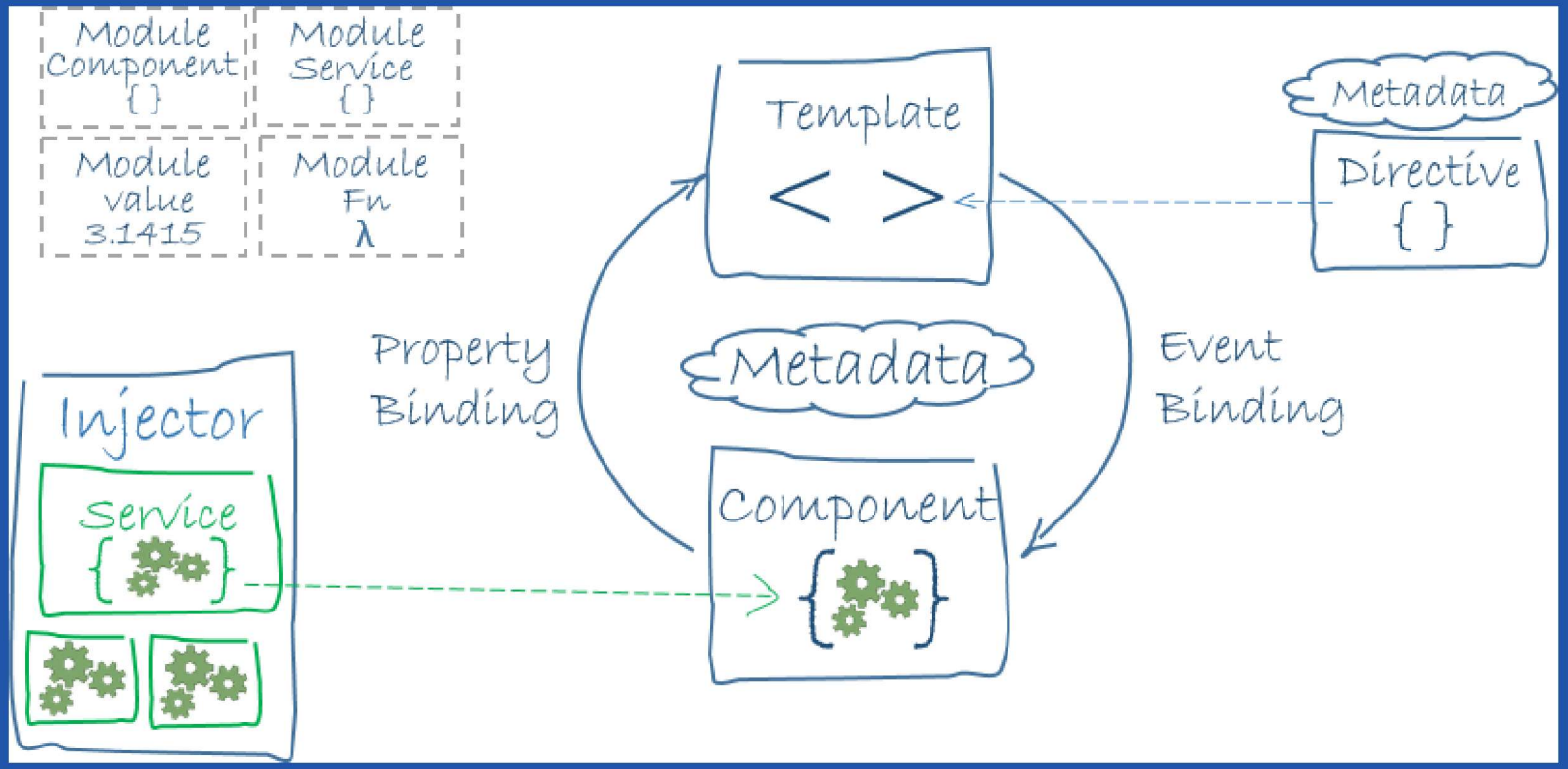

ANGULAR

Angular en quelques lignes...

- framework JavaScript
- open-source
- développé par Google
- première version en 2009 - AngularJs
- Angular 4: mars 2017

Architecture Angular

documentation officielle


Module
Component
{ }
Module
Service
{ }
Module
value
3.1415
Module
Fn
λ
Template
< >
Metadata
Directive
{ }
Property
Binding
Metadata
Event
Binding
Injector
Service
Component

L'exemple de la cave à vin

Récupération du projet

```
git clone https://github.com/jpoyard/ng2-wines-app.git
```

Installation des dépendances

```
npm i
```

Lancement du fake server

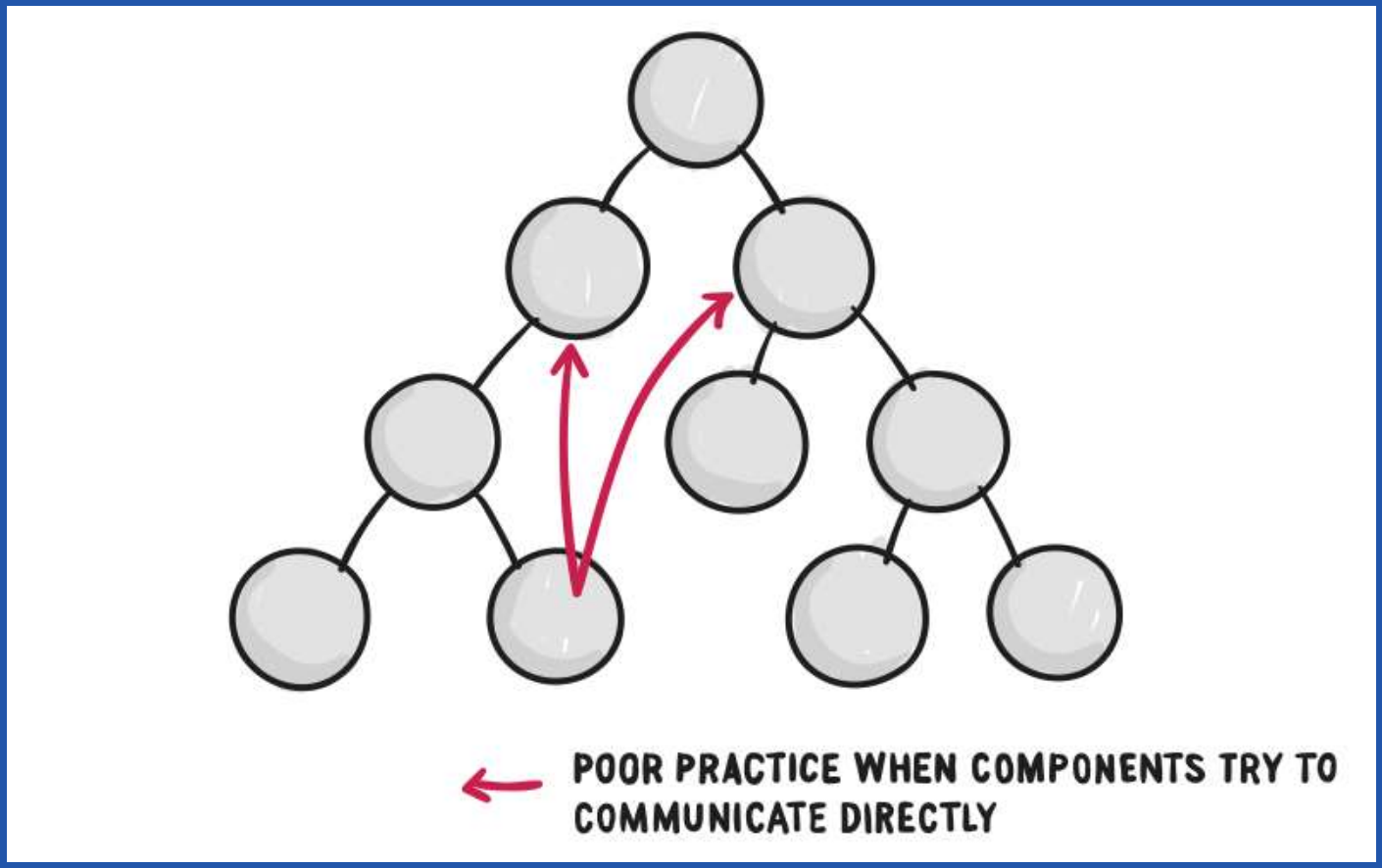
```
npm run serve
```

Lancement de webpack-dev-server

```
npm run start
```

Communication entre composants

- de parent à enfant avec **@Input**
- Intercepter les modifications avec un **setter**
- Intercepter les modifications avec **ngOnChanges()**
- en utilisant **@Output** avec **EventEmitter**
- en utilisant une **variable locale**
- le parent utilise **@ViewChild()**
- parent et enfants communiquent via un service


POOR PRACTICE WHEN COMPONENTS TRY TO
COMMUNICATE DIRECTLY

ReactiveX / RxJS

ReactiveX

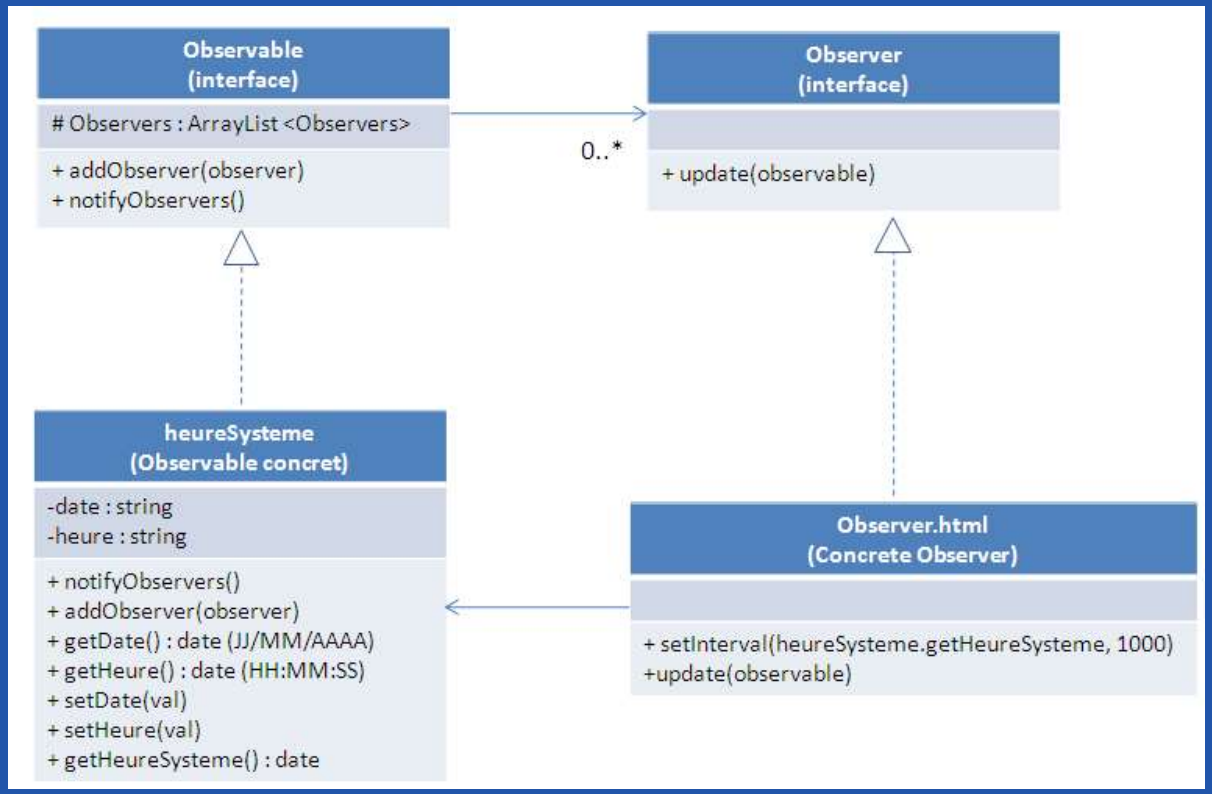
ReactiveX est une API basée sur le pattern Observer et la programmation fonctionnelle pour gérer des événements asynchrones.

RxJS

Reactive Extensions for JavaScript

pattern Observer

Le pattern Observateur (en anglais Observer) définit une relation entre objets de type un-à-plusieurs, de façon que, si un objet change d'état, tous ceux qui en dépendent en soient informés et mis à jour automatiquement.


Observable
(interface)
Observers : ArrayList <Observers>
+ addObserver(observer)
+ notifyObservers()
0..*
Observer
(interface)
+ update(observable)
heureSysteme
(Observable concret)
-date : string
-heure : string
+ notifyObservers()
+ addObserver(observer)
+ getDate() : date (JJ/MM/AAAA)
+ getHeure() : date (HH:MM:SS)
+ setDate(val)
+ setHeure(val)
+ getHeureSysteme() : date
Observer.html
(Concrete Observer)
+ setInterval(heureSysteme.getHeureSysteme, 1000)
+update(observable)

observable / subject

- Produit des données synchrone ou asynchrone
- Peut avoir un ou plusieurs observateurs
- Ne produit rien sans observateur
- Une alternative à l'utilisation des Promesses ou de EventEmitter

Les opérateurs (1/2)

- Creating Observables
- Transforming Observables
- Filtering Observables
- Combining Observables
- Error Handling Operators

Les opérateurs (2/2)

- Observable Utility Operators
- Conditional and Boolean Operators
- Mathematical and Aggregate Operators
- Backpressure Operators
- Connectable Observable Operators
- Operators to Convert Observables

Un petit exemple

```
let clickStream = Rx.Observable
  .fromEvent(document.getElementById('link'), 'click');

clickStream
  .buffer(clickStream.debounce(250))
  .map(list => list.length)
  .filter(x => x === 2)
  .subscribe(() => {
    console.log('doubleclick');
  })
```

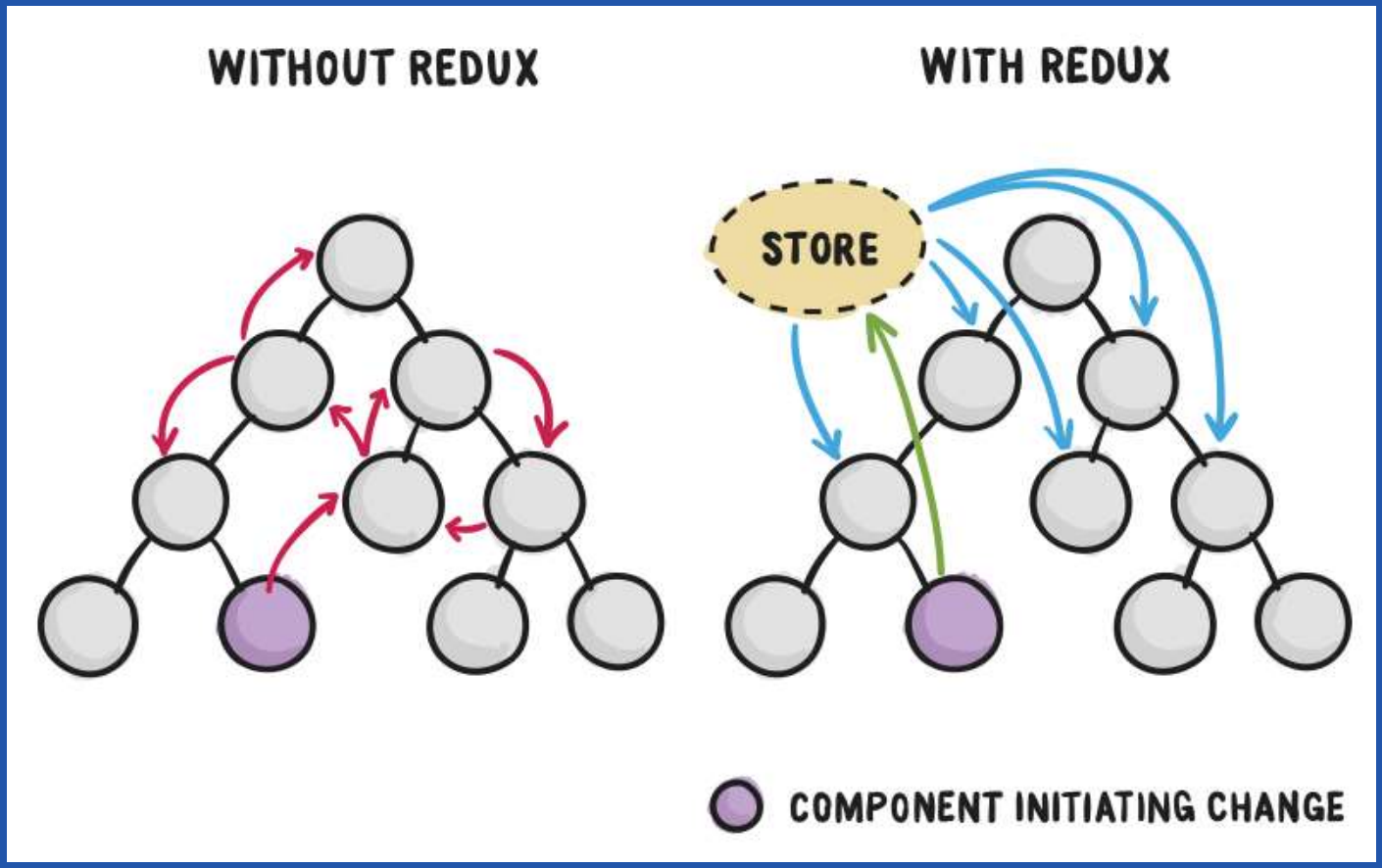
Pour aller plus loin...

- documentation ReactiveX
- Programmation Réactive Fonctionnelle (PRF)
- RxMarbles

Redux

librairie créée en Mai 2015 par Dan Abramov

"The single immutable state tree"


WITHOUT REDUX
WITH REDUX
STORE
COMPONENT INITIATING CHANGE

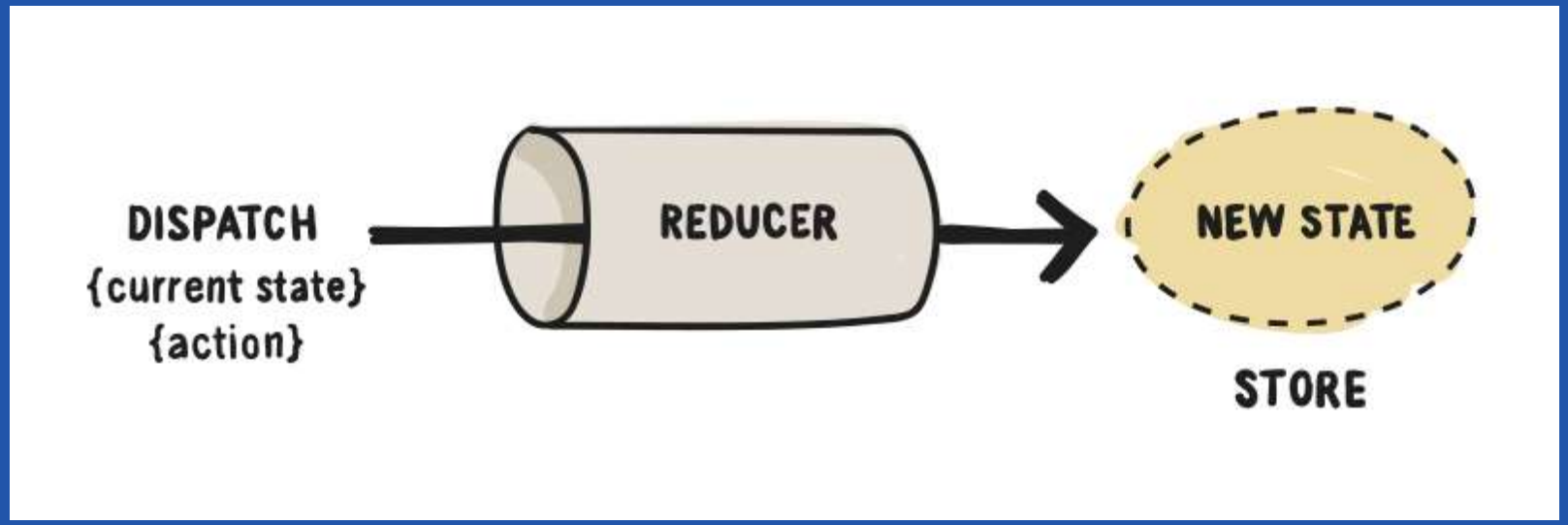
"Describing State Changes with Actions"

"Pure function vs Impure function"

```
let values = { a: 1 };

function pureFunction ( a ) {
  const b = 1;

  return a * b + 2;
}

let c = pureFunction( values.a );
```

```
let values = { a: 1 };

function impureFunction ( items ) {
  const b = 1;

  items.a = items.a * b + 2;

  return items.a;
}

let c = impureFunction( values );
```

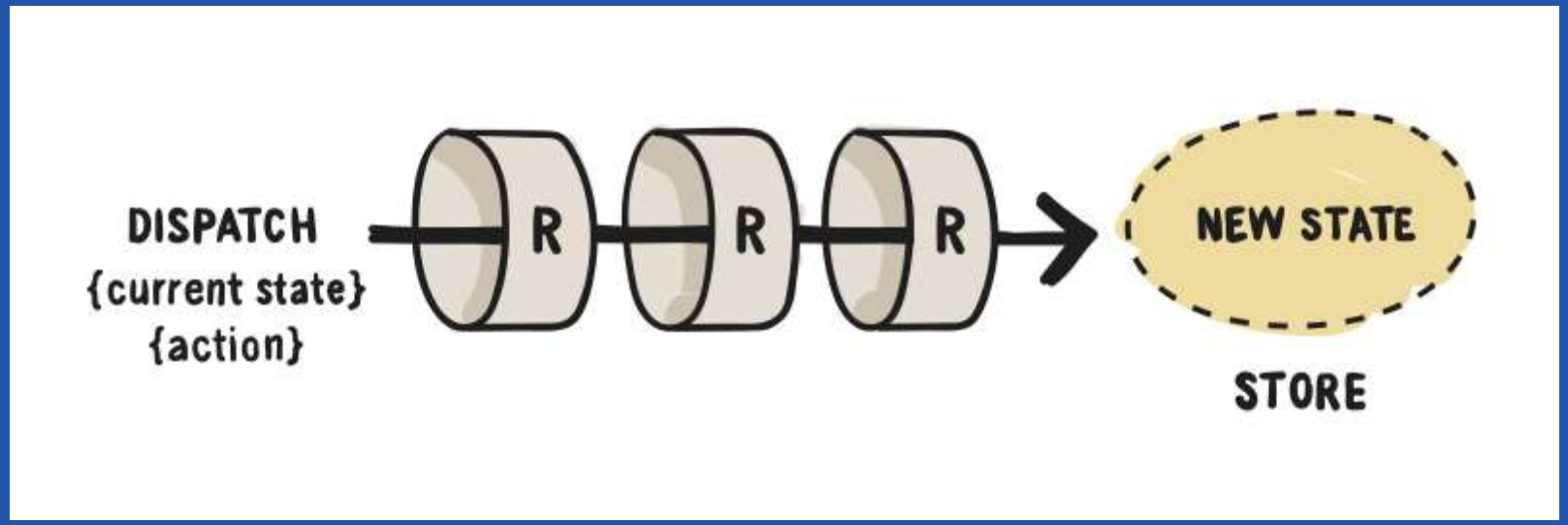
"The Reducer function"


DISPATCH
{current state}
{action}
REDUCER
NEW STATE
STORE

un exemple simple de Reducer

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

"Reducer composition with combineReducer"


DISPATCH
{current state}
{action}
R
R
R
NEW STATE
STORE

Comment marche combineReducer?

```
const combineReducers = (reducers) => {
  return (state = {}, action) => {
    return Object.keys(reducers).reduce(
      (nextState, key)=>{
        nextState[key]= reducers[key](
          state[key],
          action
        );
        return nextState;
      },
      {}
    );
  };
};
```

Combiner les reducers dans Angular

```
import { combineReducers } from 'redux';
import { composeReducers,
         defaultFormReducer } from '@angular-redux/form';
import { routerReducer } from '@angular-redux/router';
import { createWinesReducer } from '../wine/wines.reducer';

export const rootReducer = composeReducers(
  defaultFormReducer(),
  combineReducers({
    wines: createWinesReducer(),
    router: routerReducer,
}));
```

"Store"

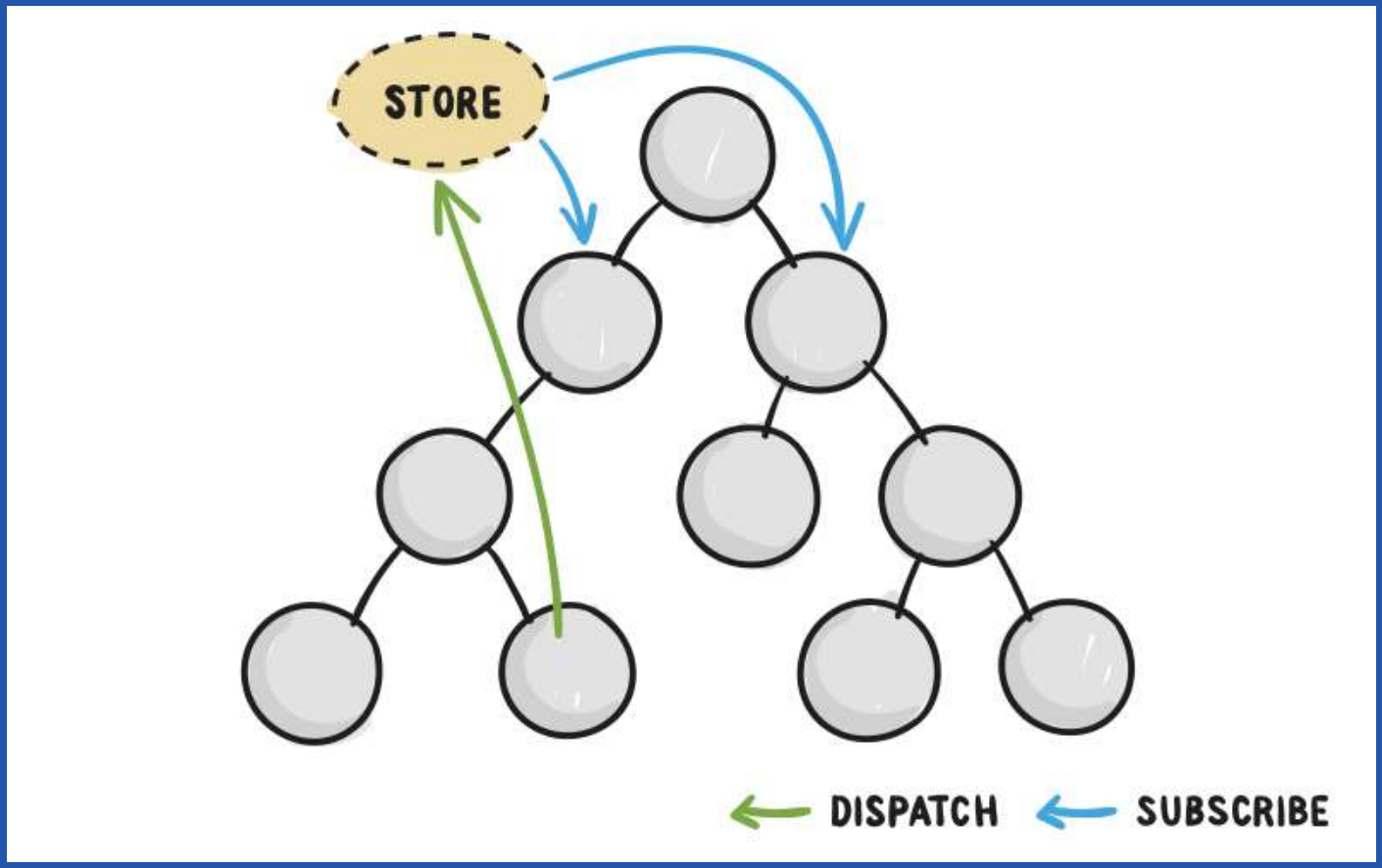
- createStore : permet la création du store
- getState() : retourne l'état
- dispatch() : réagit à une action
- subscribe() : pour souscrire à des changements

Comment marche le store?

```
const createStore = (reducer) => {
  let state; let listeners = [];
  const getState = () => state;
  const dispatch = (action) => {
    state = reducer(state, action);
    listeners.forEach(listener => listener());
  };
  const subscribe = (listener) => {
    listeners.push(listener);
    return () => {
      listeners = listeners.filter(l => l !== listener); };
  };
  dispatch({});
  return { getState, dispatch, subscribe };
}
```

Créer le store dans Angular

```
@NgModule({
  imports: [NgReduxModule, NgReduxRouterModule],
  providers: [RootEpics],
})
export class StoreModule {
  constructor(store: NgRedux<IAppState>, devTools: DevToolsExtens
    ngReduxRouter: NgReduxRouter, rootEpics: RootEpics,
  ) {
    store.configureStore(
      rootReducer, {}, [ createLogger(), ...rootEpics.createEpics
      devTools.isEnabled() ? [ devTools.enhancer() ] : []);
    ngReduxRouter.initialize();
    provideReduxForms(store);
  }
}
```


STORE
DISPATCH
SUBSCRIBE

Middlewares

“provides a third-party extension point between dispatching an action, and the moment it reaches the reducer”

Epic

" The core primitive of redux-observable is called an epic"

Maintenant c'est à vous de jouer

Utilisation de la branche *exercice*

```
git checkout exercice
```

Installation des dépendances

```
npm i
```

Lancement du fake server

```
npm run serve
```

Lancement de webpack-dev-server

```
npm run start
```

Pour aller plus loin...

Leveling Up with React: Redux

Quizz