Enhance Web Development

Angular 2

Reactive

- Le projet
- Une vue d'ensemble du Reactive
- Observable et RxJS
- Opérations Immutable
- Reactive State and @ngrx/store
- Reactive async
- Reactive Data Models

Le projet

Le projet

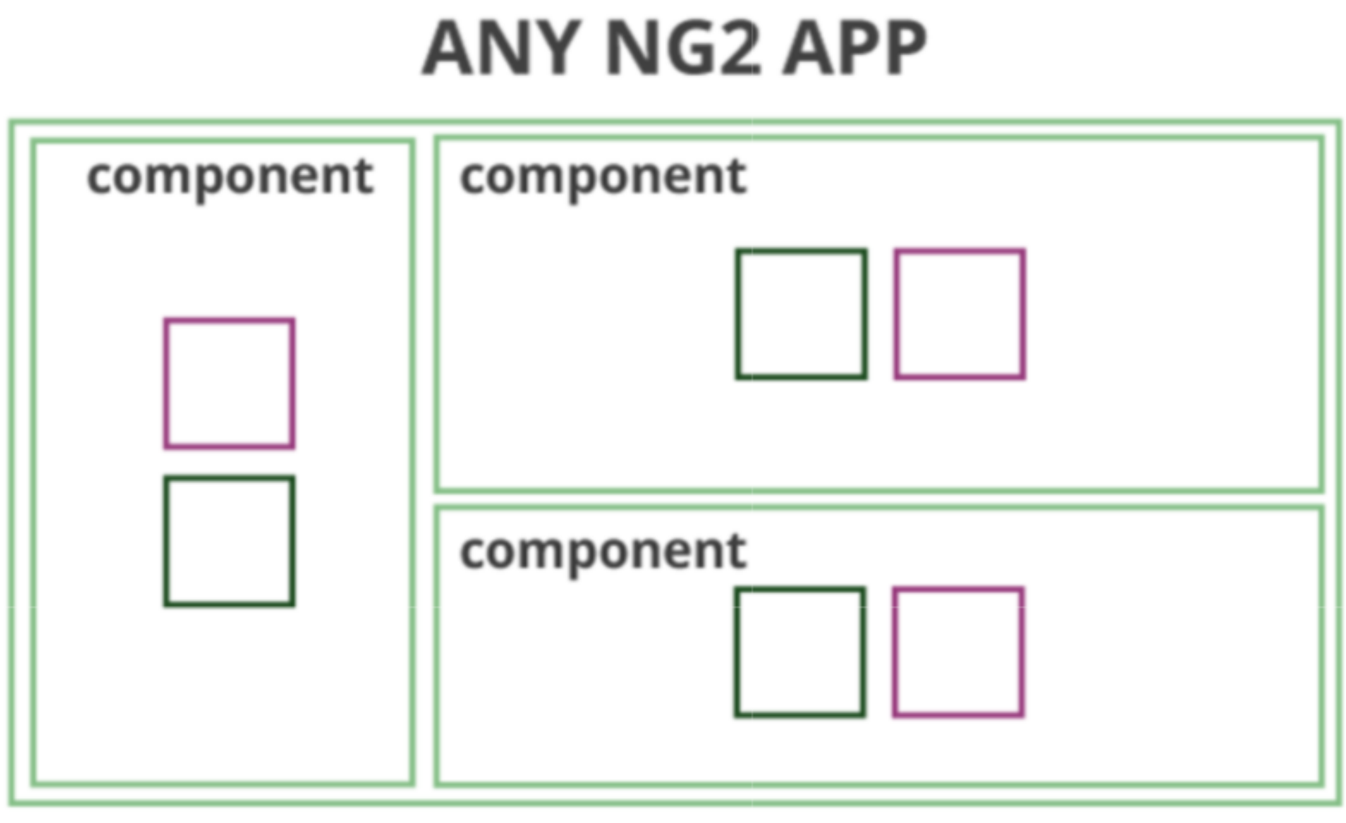
- Disponible sur la branche `ngrx`
- Met à disposition un serveur avec `json-server`
- Utilisation de `@ngrx/store`
- Transformation de **widgets** en Reactive

Une vue d'ensemble du Reactive

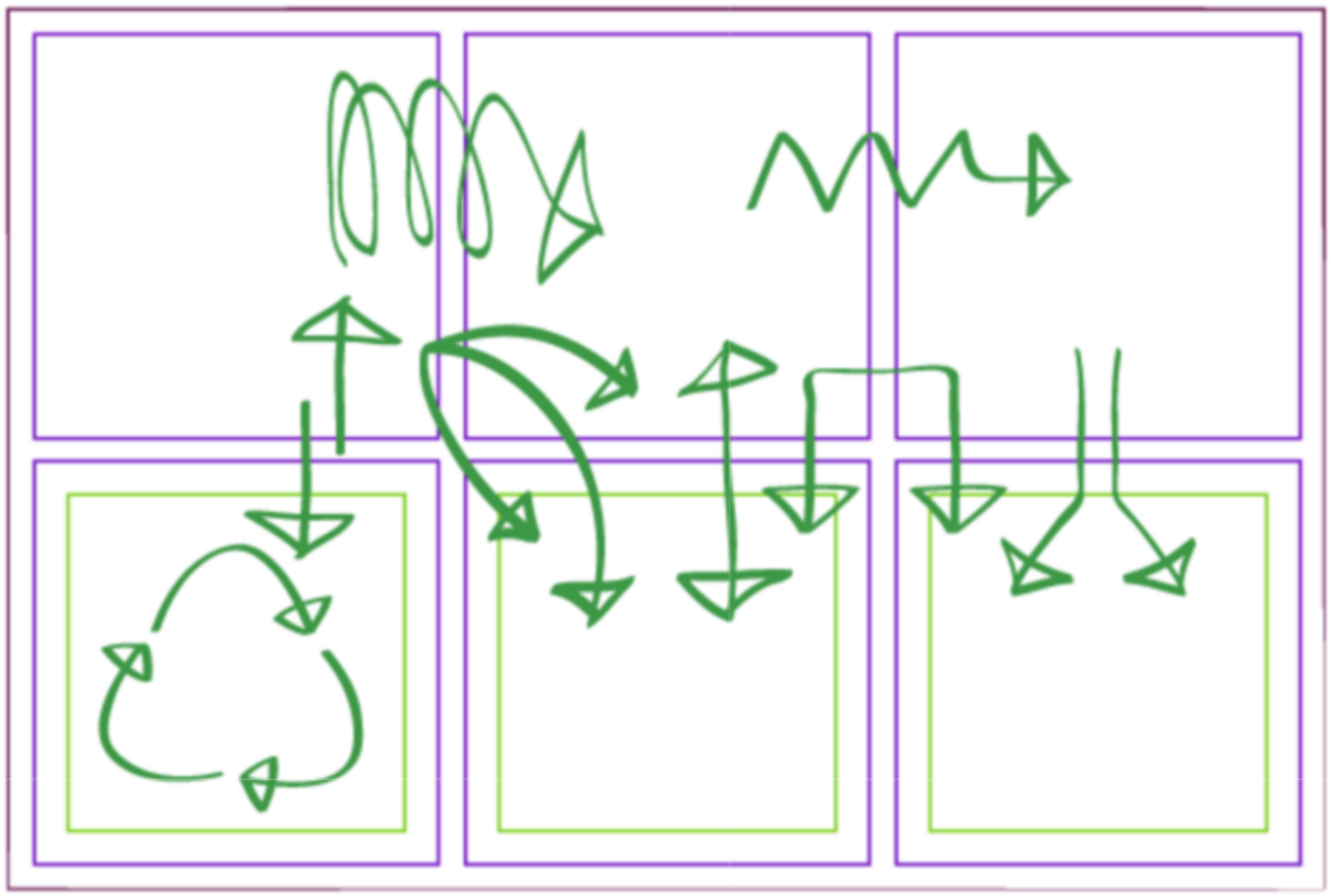
Une vue d'ensemble du Reactive

- Historique Angular
- Pourquoi Reactive ?
- Reactive Angular 2
- Découvrons Reduc

Historique Angular



Ok, petit problème !



Des etats de partout

Il faut absolument gérer ces états d'une meilleure façon

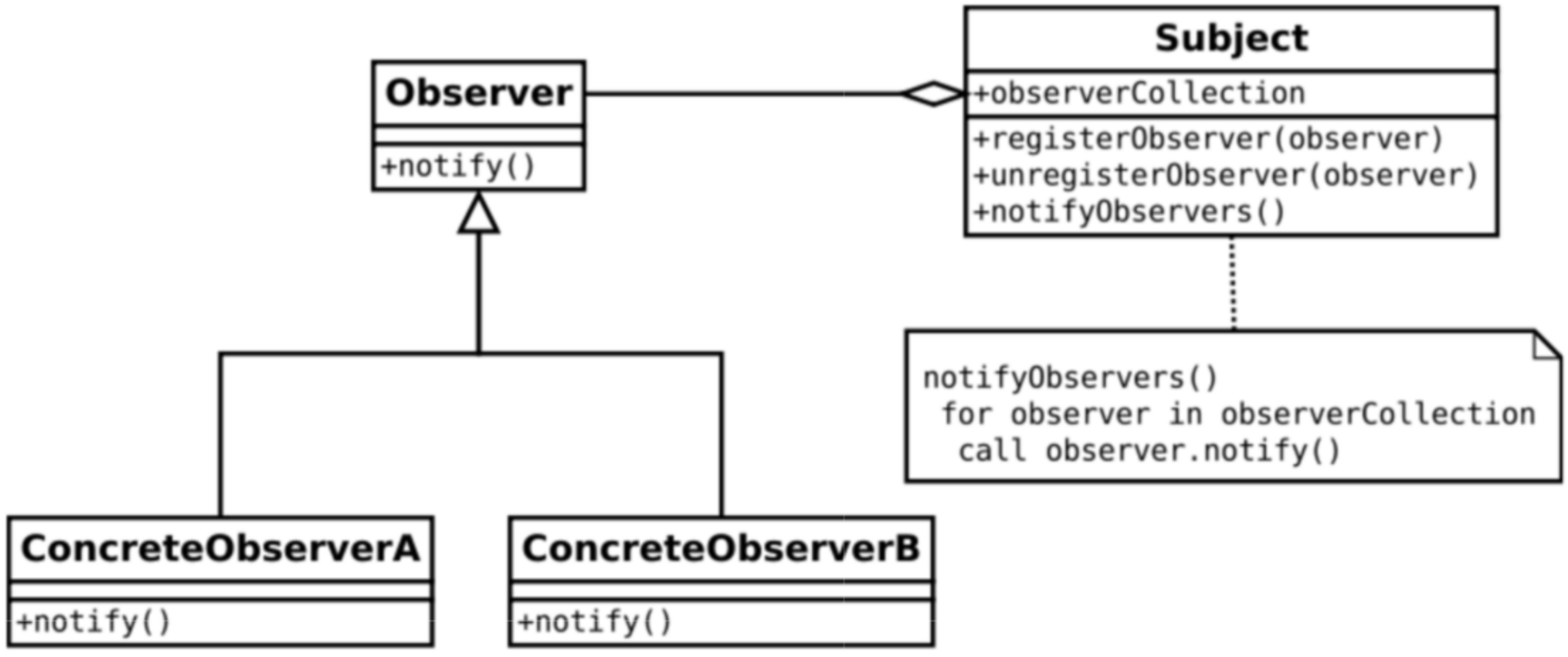
Pourquoi ne pas les pousser ces états dans notre application ?

Pourquoi ne pas simplifier la gestion de évènement utilisateurs ?

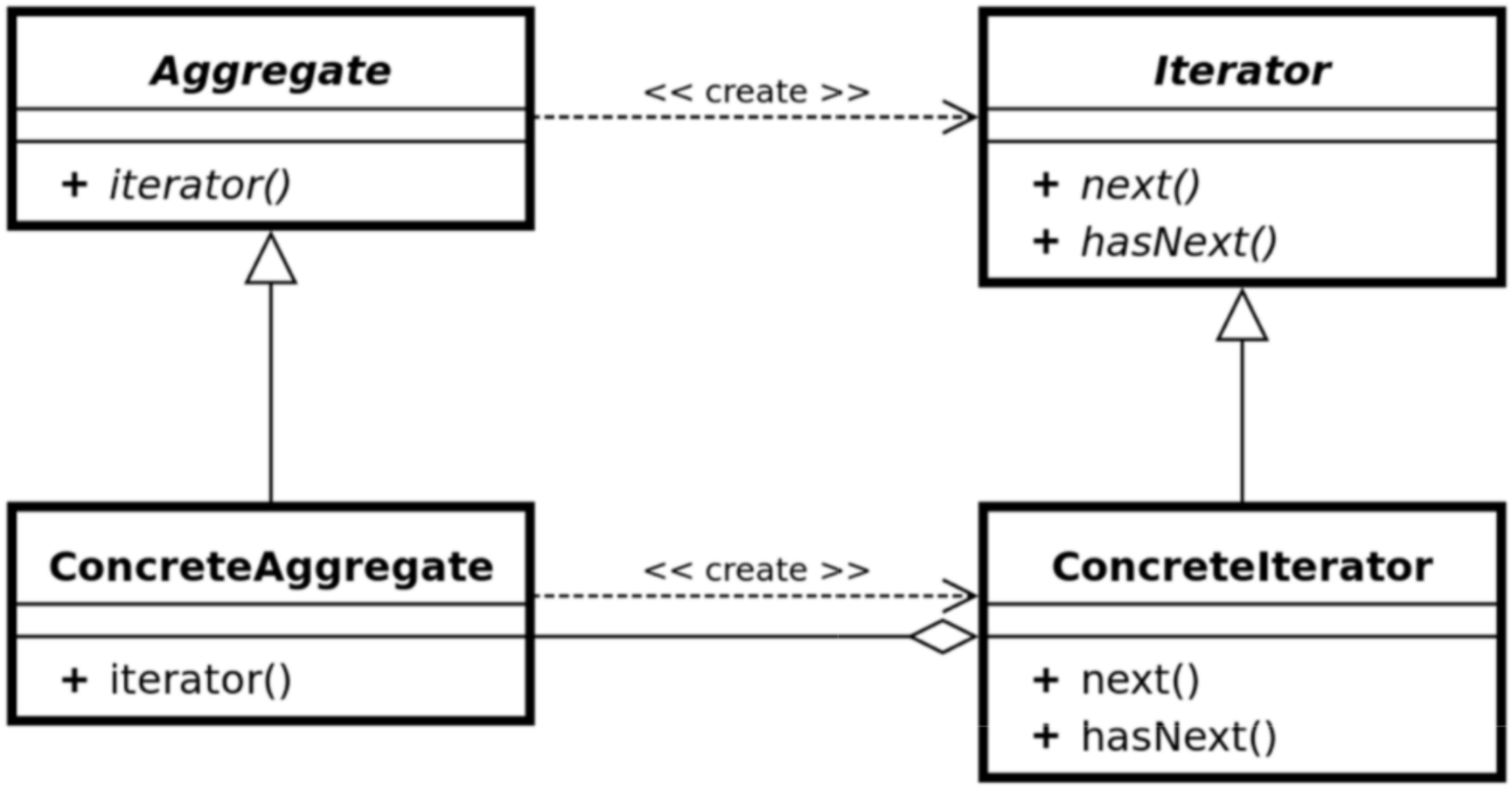
Pourquoi Reactive ?

Pourquoi Reactive ?

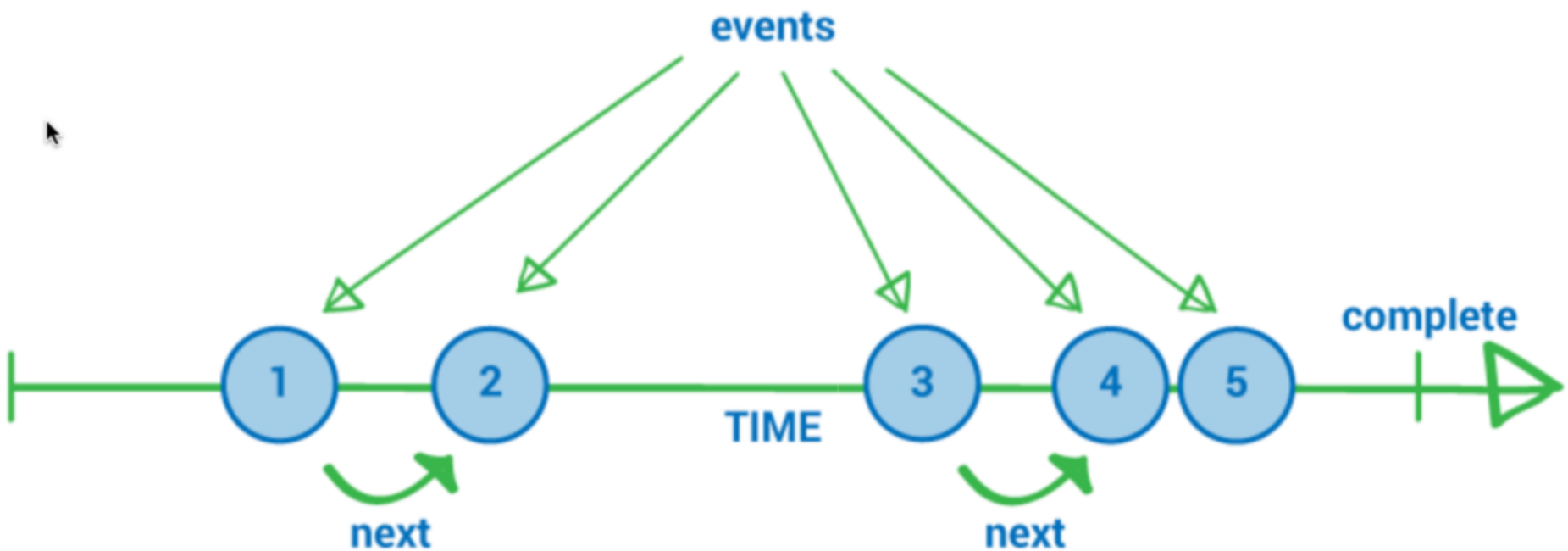
- Dans cette formation, Reactive fait référence à quand nous réagissons au données poussées sur le temps
- Utilisation des Observables Objects qui sont une extension du Pattern Observable



Observer Pattern



Iterator Pattern



Observable sequence

	SINGLE	MULTIPLE
SYNCHRONOUS	Function	Enumerable
ASYNCHRONOUS	Promise	Observable

Time + Value

	SINGLE	MULTIPLE
PULL	Function	Enumerable
PUSH	Promise	Observable

Value consumption

Reactive Angular 2

Reactive Angular 2

- Les Observables sont au coeur des modules Angular 2 (Http, Animations)
- Le pipe Async permet de les afficher facilement sans les transformer
- Observables et Immutability permettent d'effacer tout les problèmes liés à la détection de changement

```
this.http.get(BASE_URL)
  .map(res => res.json())
  .map(payload => ({ type: 'ADD_ITEMS', payload }))
  .subscribe(action => this.store.dispatch(action))
```

Observable sequence

```
<div class="mdl-cell mdl-cell--6-col">
  <items-list [items]="items | async"
              (selected)="selectItem($event)"
              (deleted)="deleteItem($event)">
  </items-list>
</div>
<div class="mdl-cell mdl-cell--6-col">
  <item-detail (saved)="saveItem($event)"
               (cancelled)="resetItem($event)"
               [item]="selectedItem | async">
    Select an Item
  </item-detail>
</div>
```

Observable sequence

Enter redux

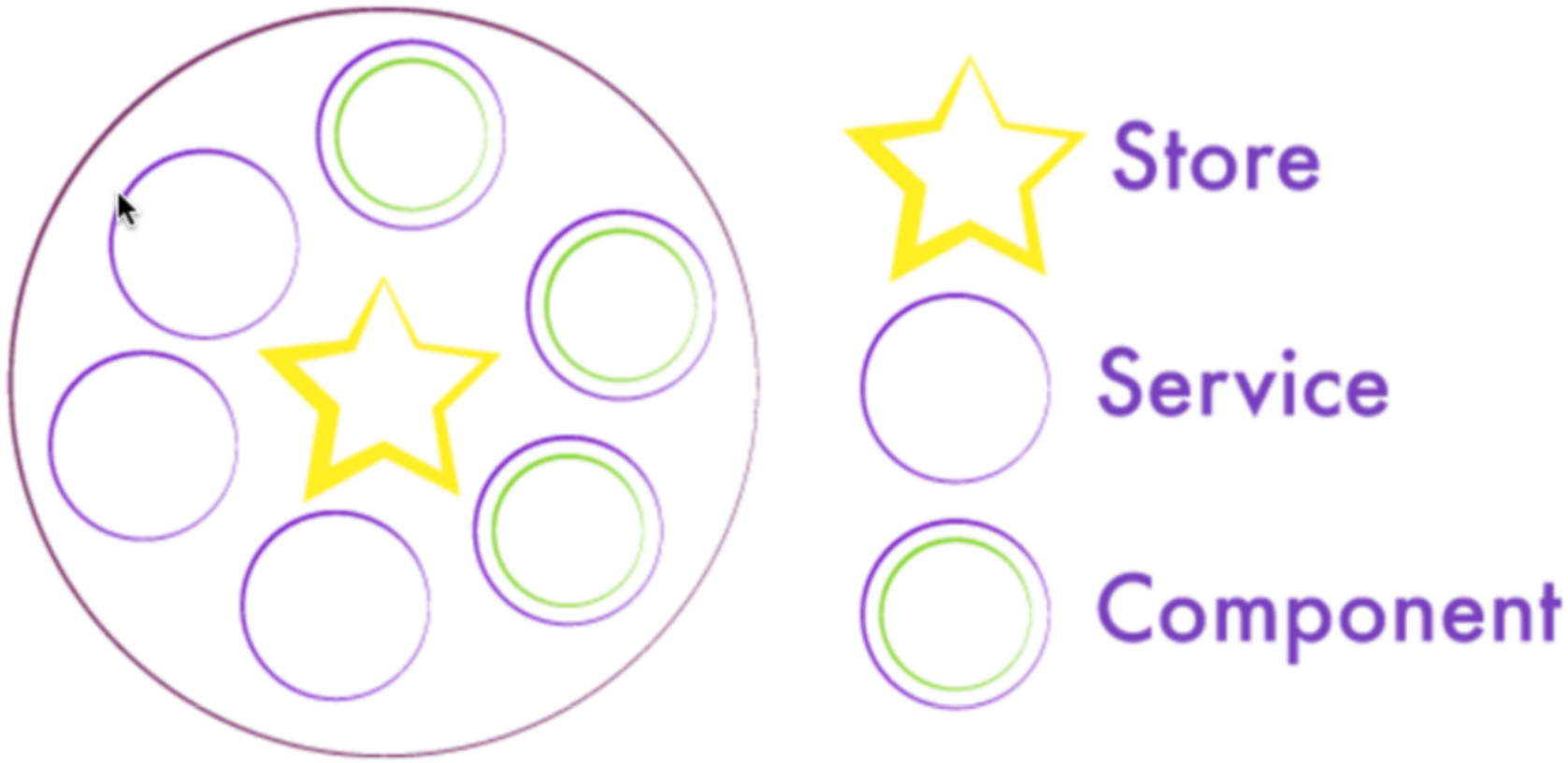
Enter Redux

- Arbre d'état immutable et unique
- Flux d'état descendant
- Flux d'évènement montant
- La gestion des états est centralisés
- Plus de découpage de logique métier dans des services ou des composants

Redux est une librairie mais plus important : c'est un pattern !

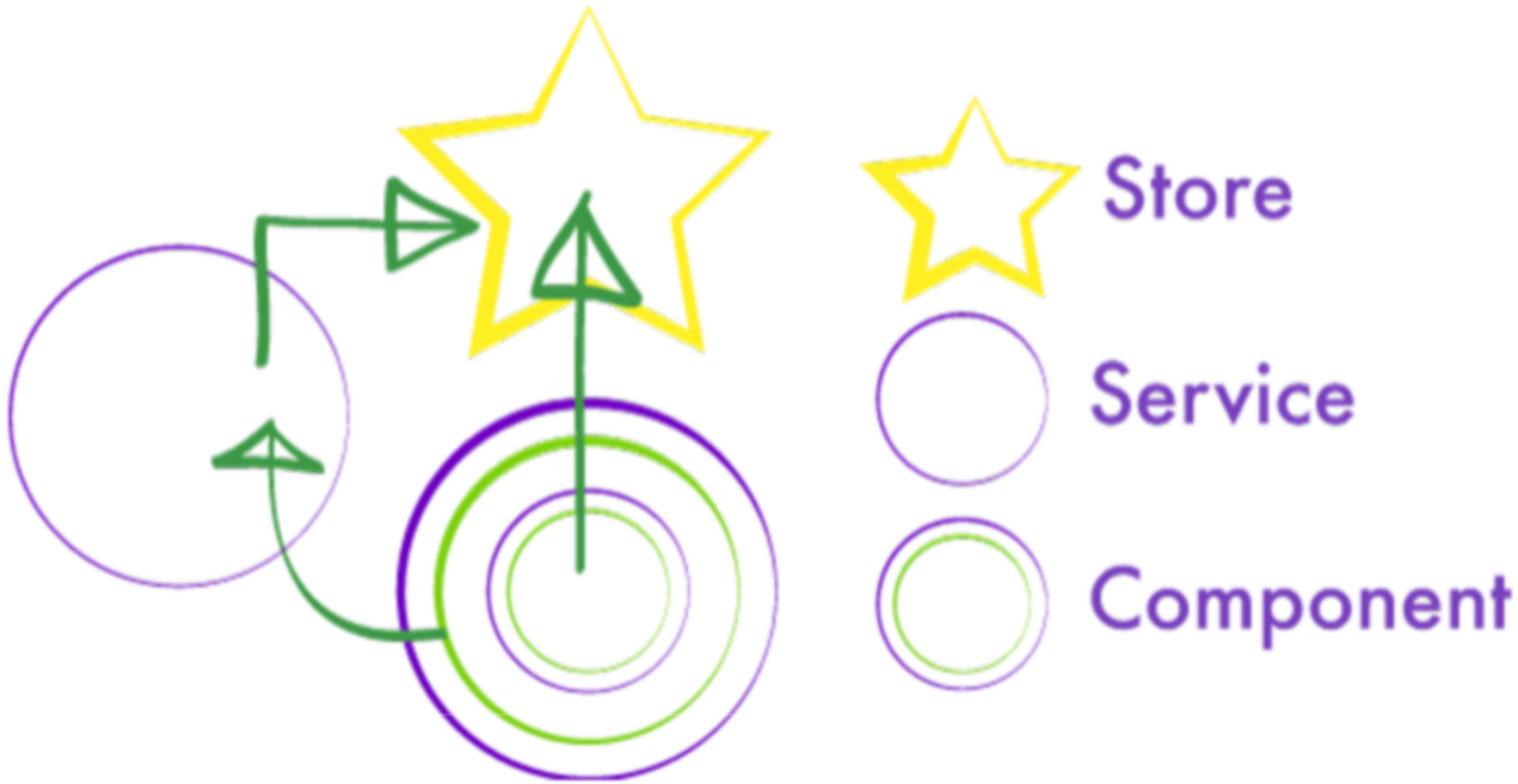
Ressources

- **Getting Started with Redux by Dan Abramov**
  https://egghead.io/courses/getting-started-with-redux
- **Reactive Angular 2 with ngrx (video)**
  https://www.youtube.com/watch?v=mhA7zZ23Odw&feature=youtu.be
- **Comprehensive Introduction to @ngrx/store**
  https://gist.github.com/btroncone/a6e4347326749f938510
- **Angular 2 - ngrx store in 10 minutes**
  https://egghead.io/lessons/angular-2-ngrx-store-in-10-minutes

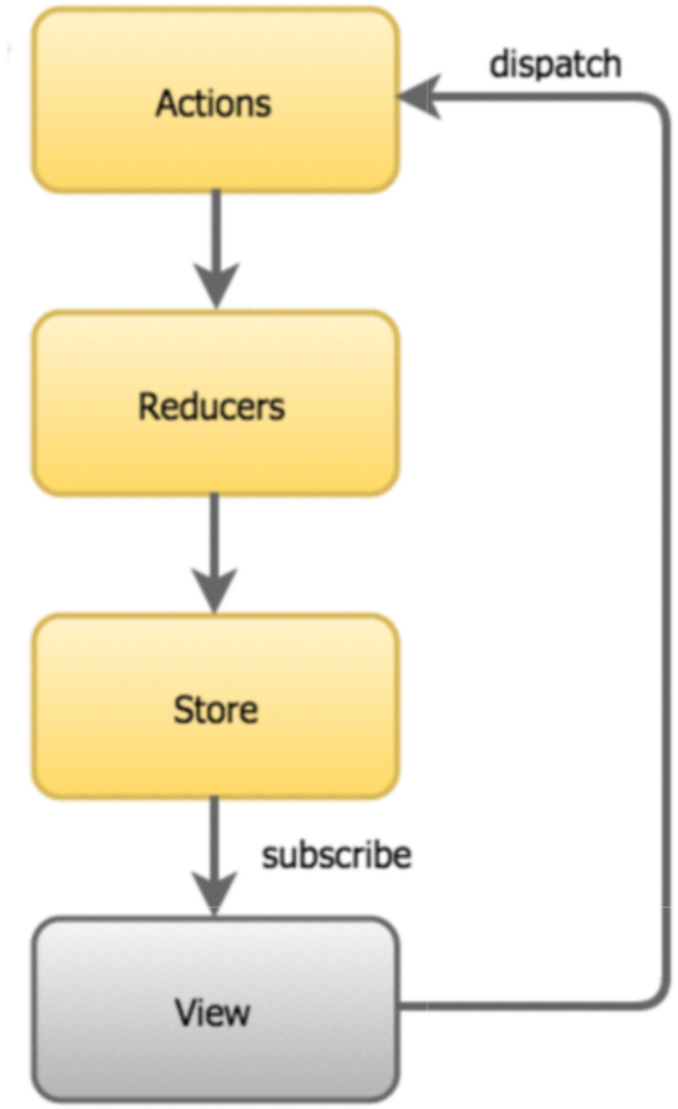


Arbre d'état unique

Flux d'état descendant

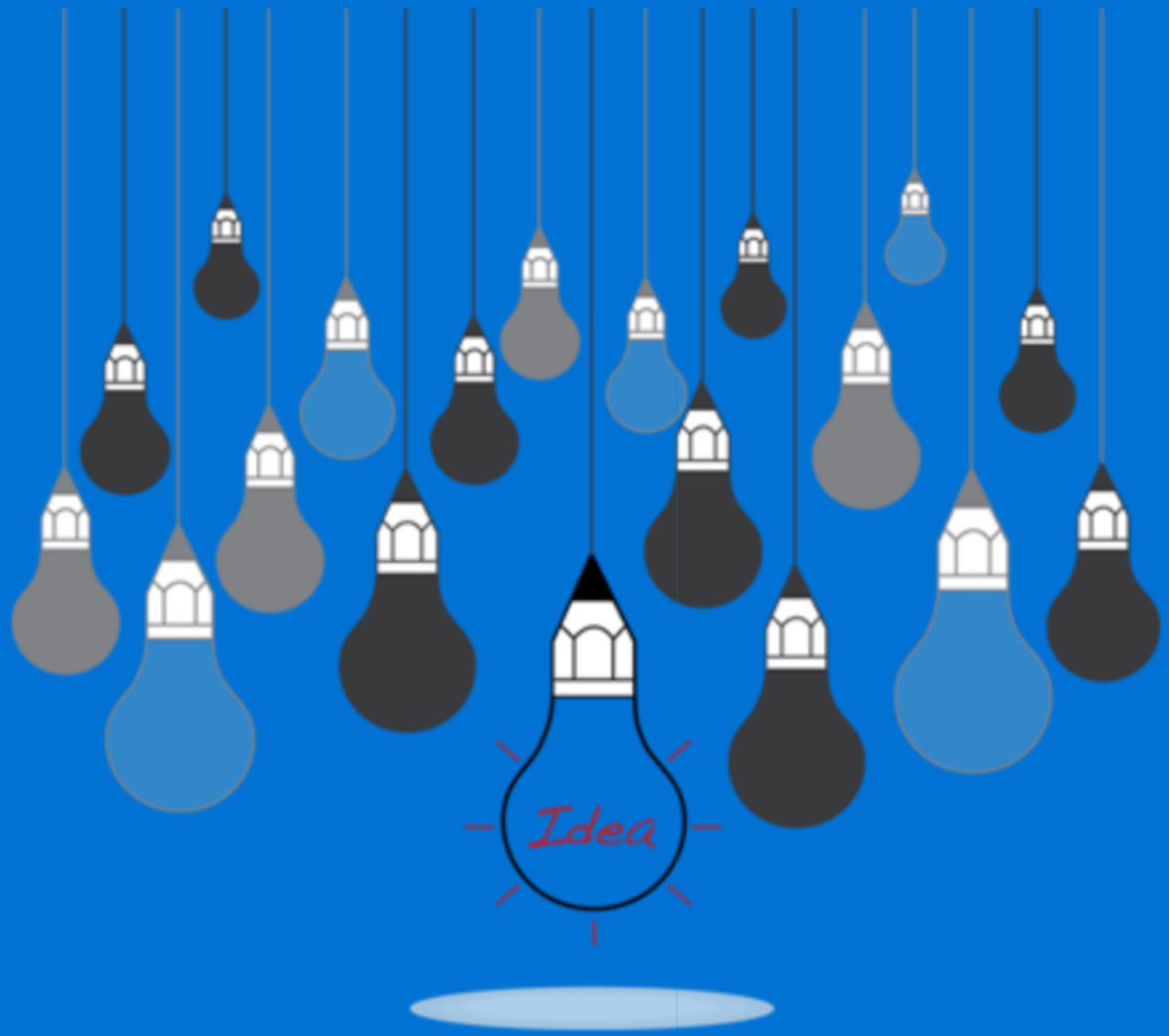


Flux d'évènement montant



Tous est reducers

Demo time !



Défi

- Télécharger et lancer l'application
- Identifier les différents concepts Reactive
- Ou sont les observables ?
- Ou sont les pipes async ?