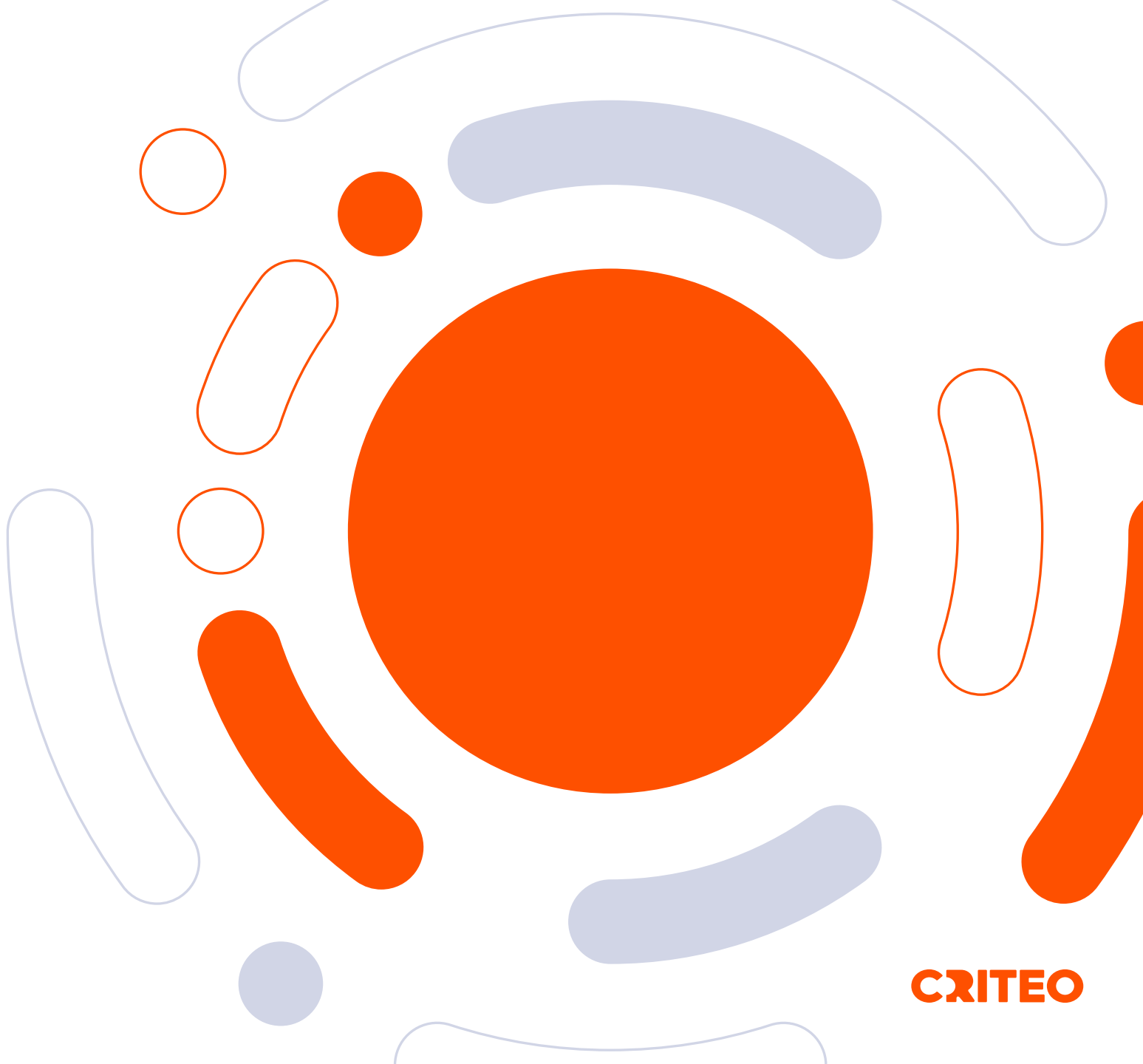


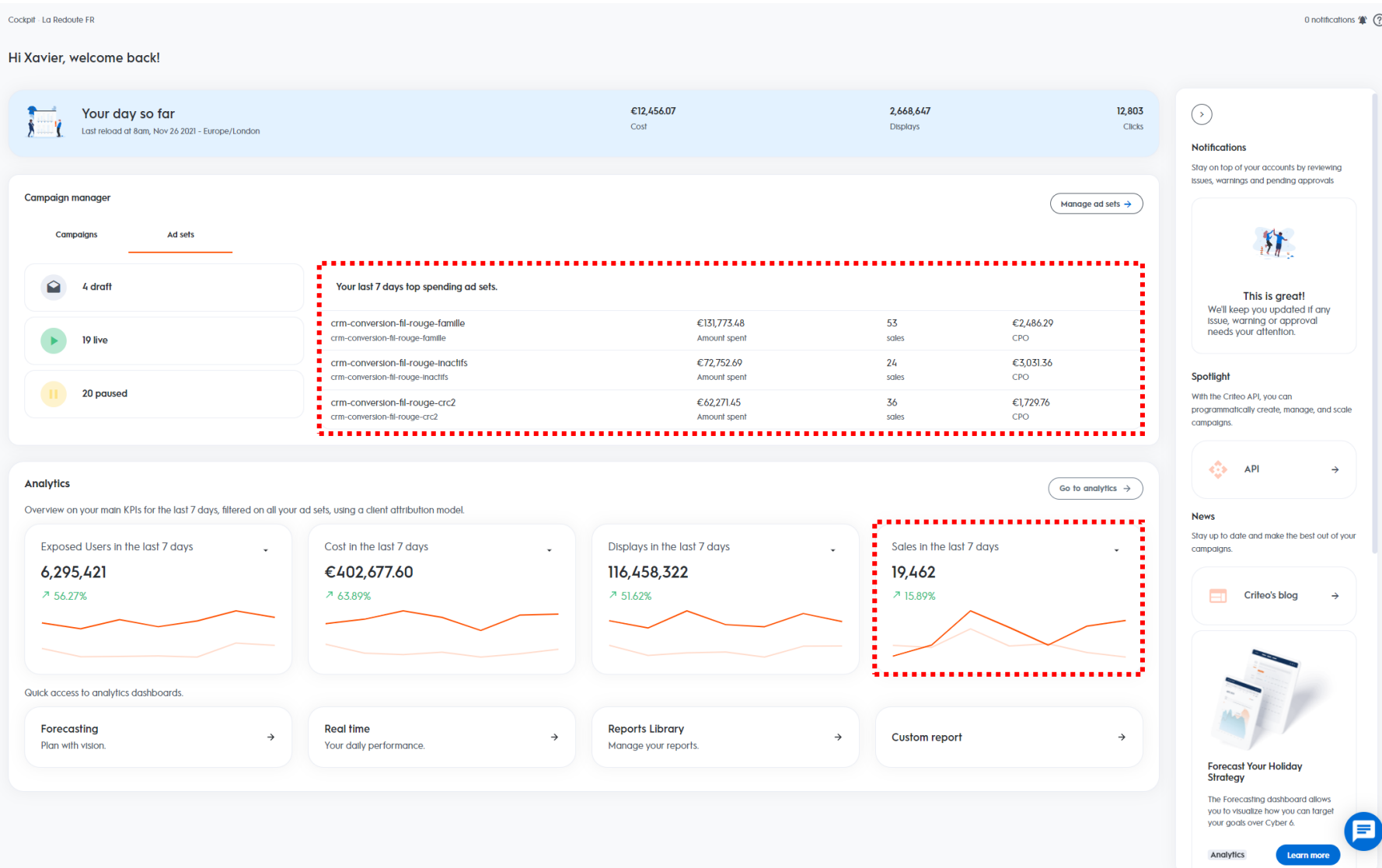
State management avec NGXS

Xavier Dupessey



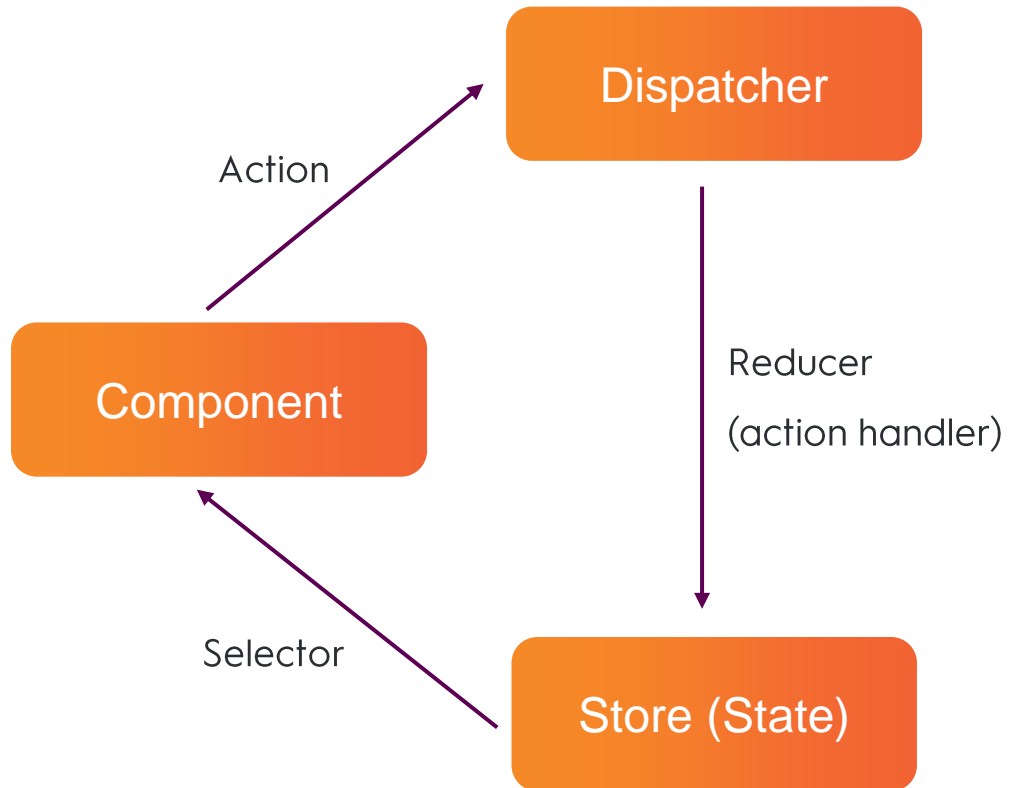
CRITEO

State management



- Communication et synchronisation
- Nombreux composants épars dans le DOM
- Contenir les responsabilités
- Faciliter les tests
- Permettre le débogage

Pattern Flux avec NGXS

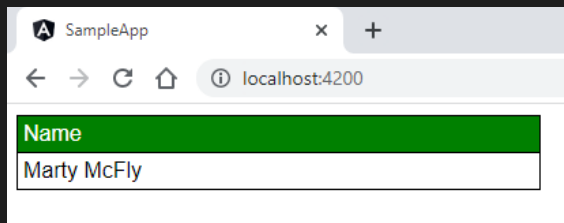


- Store immutable
- Historique
- Données normalisées
- Source de vérité
- Découple composant/logique

Implémentation avec NGXS

TS user.state.ts X

```
1 import { Injectable } from '@angular/core';
2 import { State } from '@ngxs/store';
3 import { User } from './user.model';
4
5 export interface UserStateModel {
6   users: User[];
7 }
8
9 @State<UserStateModel>({
10   name: 'user',
11   defaults: {
12     users: [{ id: 1, name: 'Marty McFly' }],
13   },
14 })
15 @Injectable()
16 export class UserState {}
17
```



TS user.selectors.ts X

```
1 import { Selector } from '@ngxs/store';
2 import { User } from './user.model';
3 import { UserState, UserStateModel } from './user.state';
4
5 export class UserSelectors {
6   @Selector([UserState])
7   static users(state: UserStateModel): User[] {
8     return state.users;
9   }
10 }
11
```

TS my-component.component.ts X

```
7 @Component({
8   selector: 'my-component',
9   template: `<table>
10     <tr>
11       <td>Name</td>
12     </tr>
13     <tr *ngFor="let user of users$ | async">
14       <td>{{ user.name }}</td>
15     </tr>
16   </table>`,
17 })
18 export class MyComponent {
19   @Select(UserSelectors.users) readonly users$: Observable<User[]>;
20 }
21
```

Implémentation avec NGXS

TS user.actions.ts X

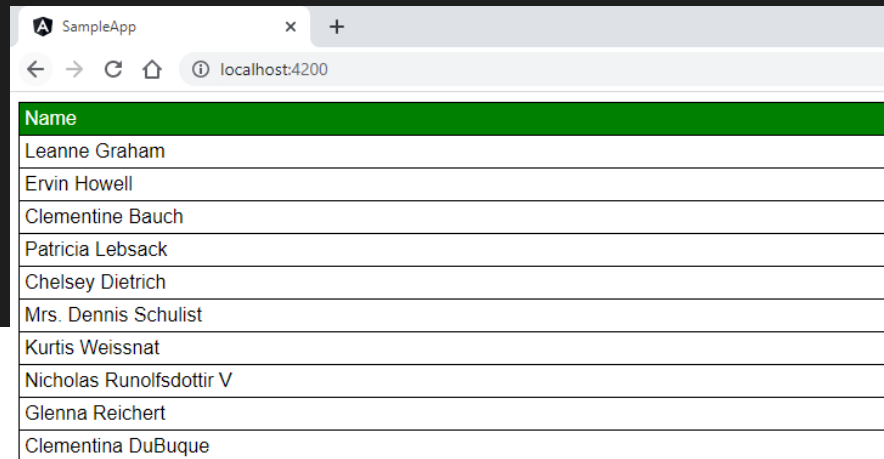
```
1 export namespace UserActions {
2   const scope = '[User]';
3
4   export class Load {
5     static readonly type = `${scope} Load`;
6   }
7 }
8 |
```

TS app.component.ts X

```
5 @Component({
6   selector: 'app-root',
7   template: `<my-component></my-component>`,
8 })
9 export class AppComponent {
10   constructor(private readonly store: Store) {
11     this.store.dispatch(new UserActions.Load());
12   }
13 }
14 |
```

TS user.state.ts X

```
12 @State<UserStateModel>({
13   name: 'user',
14   defaults: {
15     users: [],
16   },
17 })
18 @Injectable()
19 export class UserState {
20   constructor(private readonly httpClient: HttpClient) {}
21
22   @Action(UserActions.Load)
23   load({ patchState }: StateContext<UserStateModel>): Observable<unknown> {
24     return this.httpClient
25       .get<User[]>('https://jsonplaceholder.typicode.com/users')
26       .pipe(tap((users) => patchState({ users })));
27   }
28 }
29 |
```



A screenshot of a web browser window titled 'SampleApp' at 'localhost:4200'. The browser displays a table with a green header row labeled 'Name'. The table contains ten rows of names: Leanne Graham, Ervin Howell, Clementine Bauch, Patricia Lebsack, Chelsey Dietrich, Mrs. Dennis Schulist, Kurtis Weissnat, Nicholas Runolfsson, Glenna Reichert, and Clementina DuBuque.

Name
Leanne Graham
Ervin Howell
Clementine Bauch
Patricia Lebsack
Chelsey Dietrich
Mrs. Dennis Schulist
Kurtis Weissnat
Nicholas Runolfsson
Glenna Reichert
Clementina DuBuque

Debugger avec Redux DevTools

The screenshot displays the Redux DevTools interface within a web browser window titled "SampleApp" at "localhost:4200". The interface is divided into several panels:

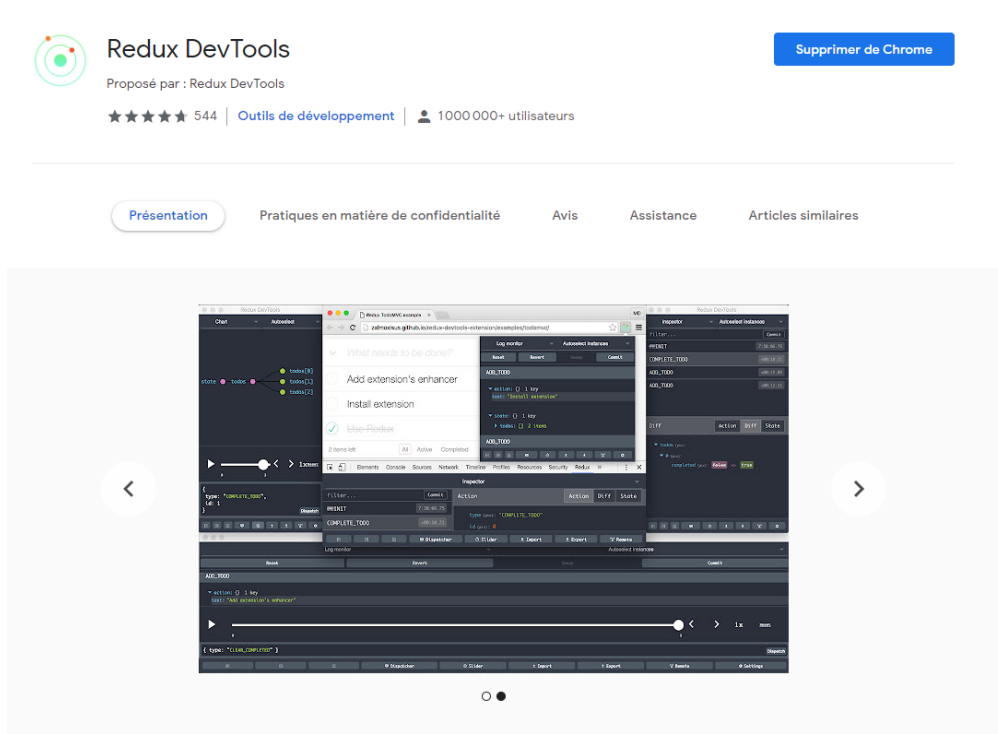
- Left Panel:** A table listing names: Leanne Graham, Ervin Howell, Clementine Bauch, Patricia Lebsack, Chelsey Dietrich, Mrs. Dennis Schulist, Kurtis Weissnat, Nicholas Runolfsdottir V, Glenna Reichert, and Clementina DuBuque.
- Inspector Panel:** Shows the Redux state tree with nodes for "filter...", "@@INIT", and "[User] Load".
- State Panel:** Displays the Redux state tree structure, showing "state" containing "user" containing "users" (an array of 10 user objects).
- Raw Panel:** Shows the raw state data for the selected "users" array, displaying a JSON object for the first user (Leanne Graham).

The raw state data for the first user is as follows:

```
{
  "address": {
    "city": "Gwenborough",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    },
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "zipcode": "92998-3874"
  },
  "company": {
    "bs": "harness real-time e-markets",
    "catchPhrase": "Multi-layered client-server neural-net",
    "name": "Romaguera-Crona"
  },
  "email": "Sincere@april.biz",
  "id": 1,
  "name": "Leanne Graham",
  "phone": "1-770-736-8031 x56442",
  "username": "Bret",
  "website": "hildegard.org"
}
```

The bottom of the interface features a timeline and a toolbar with buttons for "Pause recording", "Persist", "Dispatcher", "Slider", "Import", "Export", "Remote", and "Settings".

Debugger avec Redux DevTools



Présentation

Compatible avec votre appareil

Redux DevTools for debugging application's state changes.

The extension provides power-ups for your Redux development workflow. Apart from Redux, it can be used with any other architectures which handle the state.

This is an open source project. See the official repository for more details: <https://github.com/reduxjs/redux-devtools>

Informations supplémentaires

[Site Web](#) [Signaler un abus](#)

Version

2.17.2

Dernière mise à jour

1 juin 2021

Taille

3.71MiB

Langue

English

Développeur

[Contacter le développeur](#)

```
TS app.module.ts M X
1 import { HttpClientModule } from '@angular/common/http';
2 import { NgModule } from '@angular/core';
3 import { BrowserModule } from '@angular/platform-browser';
4 import { NgxsReduxDevtoolsPluginModule } from '@ngxs/devtools-plugin';
5 import { NgxsModule, NoopNgxsExecutionStrategy } from '@ngxs/store';
6 import { environment } from 'src/environments/environment';
7 import { AppComponent } from './app.component';
8 import { MyComponent } from './my-component/my-component.component';
9 import { UserState } from './user-state/user.state';
10
11 @NgModule({
12   declarations: [AppComponent, MyComponent],
13   imports: [
14     NgxsReduxDevtoolsPluginModule.forRoot({
15       disabled: environment.production,
16       maxAge: 50,
17     }),
18     BrowserModule,
19     HttpClientModule,
20     NgxsModule.forRoot([UserState], {
21       developmentMode: !environment.production,
22       executionStrategy: NoopNgxsExecutionStrategy,
23       selectorOptions: { injectContainerState: false, suppressErrors: false },
24     }),
25   ],
26   bootstrap: [AppComponent],
27 })
28 export class AppModule {}
29
```

Test unitaire

TS my-component.component.spec.ts X

Run | Debug

```
7 describe('MyComponent', () => {
8   let component: MyComponent;
9
10  beforeEach(() => {
11    TestBed.configureTestingModule({
12      imports: [NgxsModule.forRoot([UserState]), HttpClientTestingModule],
13      declarations: [MyComponent],
14    }).overrideComponent(MyComponent, { set: { template: '' } });
15
16    component = TestBed.createComponent(MyComponent).componentInstance;
17
18    const userInitialState: UserStateModel = {
19      users: [{ id: 1, name: 'Marty McFly' }],
20    };
21
22    TestBed.inject(Store).reset({ user: userInitialState });
23  });
24
25  Run | Debug
26  it('should get users', (done) => {
27    component.users$.subscribe((users) => {
28      expect(users).toEqual([{ id: 1, name: 'Marty McFly' }]);
29      done();
30    }, done.fail);
31  });
32  });
```

~ même principe pour les tests automatisés UI (par exemple avec Cypress) !

<https://docs.cypress.io/guides/references/best-practices>

Simple state, smart selectors

Comment obtenir l'âge à partir de la date de naissance ?

- Dans le composant ?
- Dans le store ?
- Dans le sélecteur !

Name	Birthdate	Age
Leanne Graham	Jan 14, 1980	41
Ervin Howell	Aug 17, 1958	63
Clementine Bauch	Nov 11, 1958	63
Patricia Lebsack	May 23, 1992	29
Chelsey Dietrich	Aug 11, 1969	52
Mrs. Dennis Schulist	Feb 25, 2016	5
Kurtis Weissnat	Nov 20, 1951	70
Nicholas Runolfsdottir V	Oct 18, 2017	4
Glenna Reichert	Jul 22, 1953	68
Clementina DuBuque	May 13, 1957	64

TS user.selectors.ts X

```
1 import { Selector } from '@ngxs/store';
2 import { User } from '../user.model';
3 import { UserState, UserStateModel } from '../user.state';
4
5 export class UserSelectors {
6   @Selector([UserState])
7   static users(state: UserStateModel): User[] {
8     return state.users.map((user) => ({
9       ...user,
10      age: this.age(user.birthDate),
11    }));
12   }
13
14   private static age(birthDate: Date): number {
15     const ageDifMs = Date.now() - birthDate.getTime();
16     const ageDate = new Date(ageDifMs);
17     return Math.abs(ageDate.getUTCFullYear() - 1970);
18   }
19 }
20
```

Simple state, smart selectors

TS user.state.ts X

```
10 export interface UserStateModel {
11   premiumIds: number[];
12   users: User[];
13 }
14
15 @State<UserStateModel>({
16   name: 'user',
17   defaults: {
18     premiumIds: [1, 2, 3], ←
19     users: [],
20   },
21 })
22 @Injectable()
23 export class UserState {
24   constructor(private readonly httpClient: HttpClient) {}
25
26   @Action(UserActions.Load)
27   load({ patchState }: StateContext<UserStateModel>): Observable<unknown> {
28     return this.httpClient
29       .get<User[]>('https://isoonplaceholder.tynicode.com/users')
```

TS my-component.component.ts X

```
22 export class MyComponent {
23   @Select(UserSelectors.premium) readonly users$: Observable<User[]>;
24 }
25
```

TS user.selectors.ts X

```
5 export class UserSelectors {
6   @Selector([UserState])
7   static premium(state: UserStateModel): User[] {
8     const premium = new Set(state.premiumIds);
9     return state.users.filter((user) => premium.has(user.id));
10  }
11 }
```

SampleApp x +

localhost:4200

Id	Name	Birthdate
1	Leanne Graham	Feb 14, 2013
2	Ervin Howell	Nov 12, 2020
3	Clementine Bauch	Feb 18, 1961

Composition de sélecteurs

TS user.selectors.ts X

```
5 export class UserSelectors {
6   @Selector([UserState])
7   static premiumIdsSlice(state: UserStateModel): number[] {
8     return state.premiumIds;
9   }
10
11   @Selector([UserSelectors.premiumIdsSlice])
12   static premiumIdsSet(premiumIdsSlice: number[]): Set<number> {
13     return new Set(premiumIdsSlice);
14   }
15
16   @Selector([UserSelectors.premiumIdsSet, UserSelectors.usersSlice])
17   static premium(premiumIdsSet: Set<number>, usersSlice: User[]): User[] {
18     return usersSlice.filter((user) => premiumIdsSet.has(user.id));
19   }
20
21   @Selector([UserState])
22   static usersSlice(state: UserStateModel): User[] {
23     return state.users;
24   }
25 }
26
```

t0

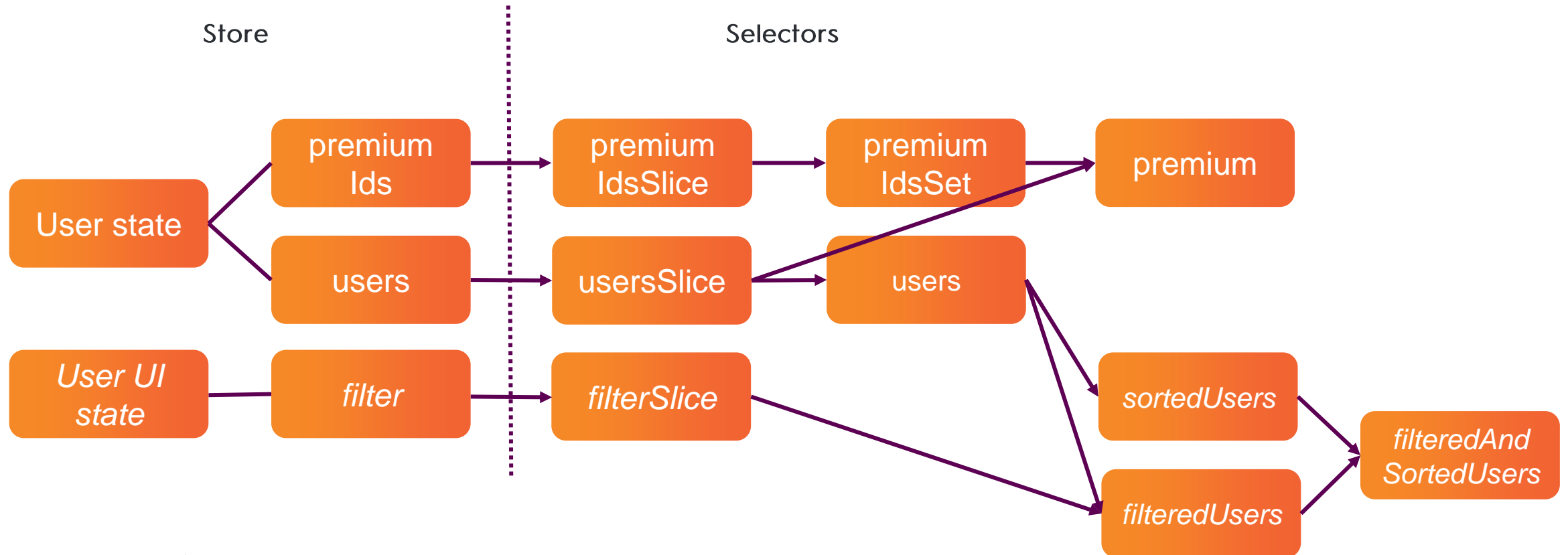
- 3 Retourne [1, 2, 3] (ref0)
- 2 Retourne Set([1, 2, 3]) (ref1)
- 1 Retourne [] (ref3)
- 4 Retourne [] (ref2)

t1

- 3 Retourne [1, 2, 3] (ref0)
- 2 Non exécutée (ref1)
- 1 Retourne [{ user1 }, ..., { user3 }] (ref5)
- 4 Retourne [{ user1 }, ..., { user10 }] (ref4)

patchState({ users })

Performances



- Arbre de dépendance entre sélecteurs
- Résultats des sélecteurs gardés en **mémoire** et réutilisés

Configuration

```
TS app.module.ts M X
11 @NgModule({
12   imports: [
13     NgxsModule.forRoot([UserState], {
14       developmentMode: !environment.production,
15       executionStrategy: NoopNgxsExecutionStrategy,
16       selectorOptions: { injectContainerState: false, suppressErrors: false },
17     }),
18     NgxsReduxDevtoolsPluginModule.forRoot({
19       disabled: environment.production,
20       maxAge: 50,
21     }),
22     BrowserModule,
23     HttpClientModule,
24   ],
25   declarations: [AppComponent, MyComponent],
26   providers: [],
27   bootstrap: [AppComponent],
28 })
29 export class AppModule {}
```

Sélecteurs paramétrés

TS user.selectors.ts X

```
5 export type OverAgeFn = (age: number) => User[];
6
7 export class UserSelectors {
8   @Selector([UserState])
9   static overAgeFn(state: UserStateModel): OverAgeFn {
10     return (age: number) =>
11       state.users
12         .map((user) => ({
13           ...user,
14           age: this.age(user.birthDate),
15         })))
16         .filter((user) => user.age > age);
17   }
18
19   private static age(birthDate: Date): number {
20     const ageDifMs = Date.now() - birthDate.getTime();
21     const ageDate = new Date(ageDifMs);
22     return Math.abs(ageDate.getUTCFullYear() - 1970);
23   }
24 }
25
26 @Selector([UserSelectors.overAgeFn])
27 static over25(overAgeFn: OverAgeFn): User[] {
28   return overAgeFn(25);
29 }
```

TS my-component.component.ts X

```
23 export class MyComponent {
24   over25: User[] = [];
25
26   constructor(private readonly store: Store) {
27     this.store.select(UserSelectors.overAgeFn).subscribe((overAgeFn) => {
28       this.over25 = overAgeFn(25);
29     });
30   }
31 }
32
```

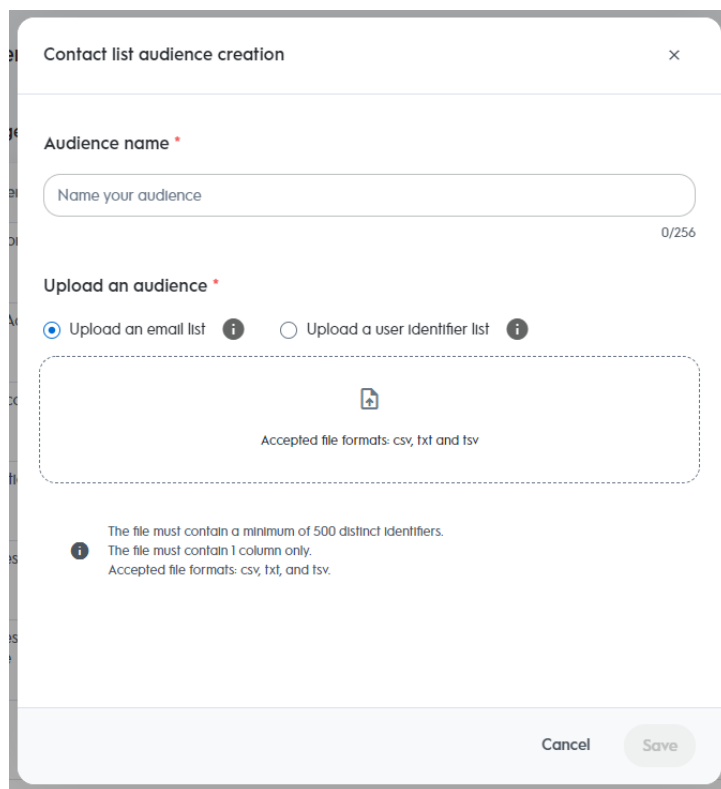
SampleApp X +

localhost:4200

Id	Name	Birthdate	Age
2	Ervin Howell	May 2, 1968	53
5	Chelsey Dietrich	Jun 25, 1972	49
7	Kurtis Weissnat	Dec 15, 1982	38
8	Nicholas Runolfsdottir V	May 19, 1957	64
9	Glenna Reichert	Jun 24, 1970	51

To be, or not to be (in the state)

« ça dépend »



Contact list audience creation

Audience name *

Name your audience 0/256

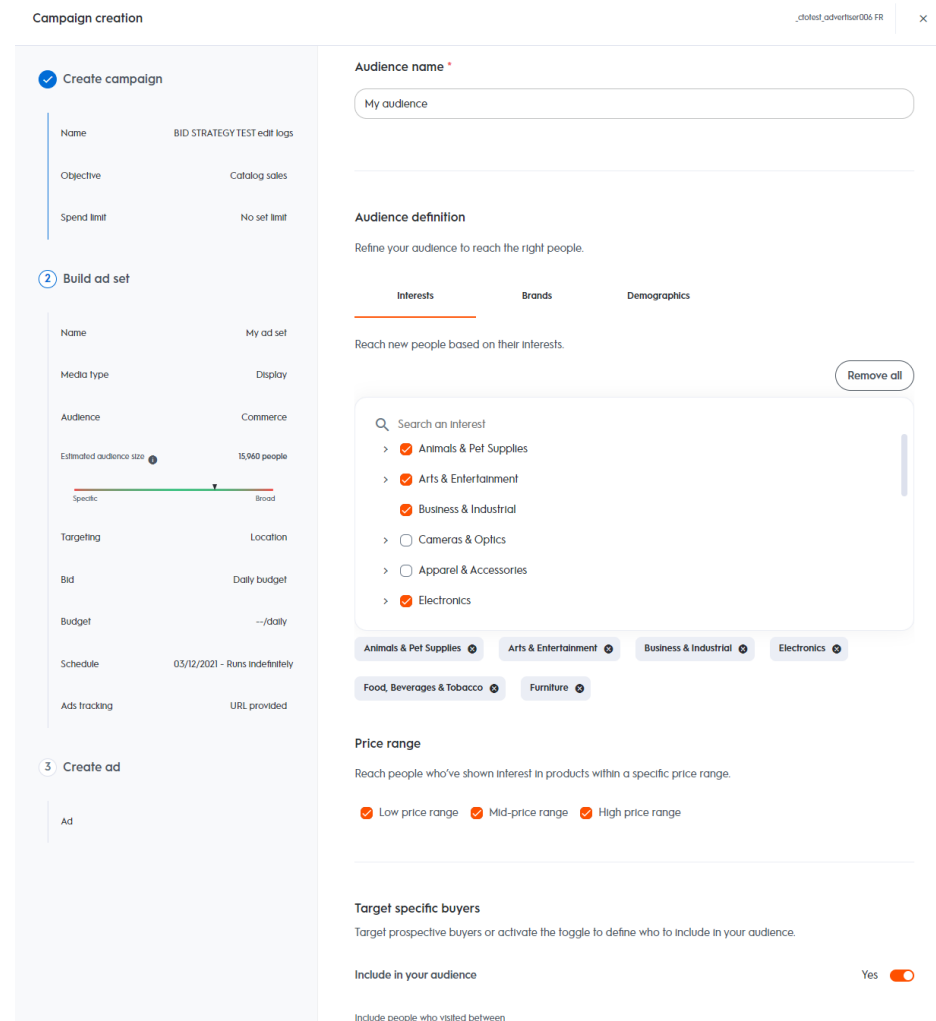
Upload an audience *

☒ Upload an email list *i* ☐ Upload a user identifier list *i*

Accepted file formats: csv, txt and tsv

i The file must contain a minimum of 500 distinct identifiers.
The file must contain 1 column only.
Accepted file formats: csv, txt, and tsv.

Cancel Save



Campaign creation

Create campaign

Name: BID STRATEGY TEST edit logs
Objective: Catalog sales
Spend limit: No set limit

Build ad set

Name: My ad set
Media type: Display
Audience: Commerce
Estimated audience size: 15,060 people
Targeting: Location
Bid: Daily budget
Budget: --/daily
Schedule: 03/12/2021 - Runs indefinitely
Ads tracking: URL provided

Create ad

Ad

Audience name *

My audience

Audience definition

Refine your audience to reach the right people.

Interests Brands Demographics

Reach new people based on their interests.

Remove all

Search an interest

- ☒ Animals & Pet Supplies
- ☒ Arts & Entertainment
- ☒ Business & Industrial
- ☐ Cameras & Optics
- ☐ Apparel & Accessories
- ☒ Electronics

Animals & Pet Supplies Arts & Entertainment Business & Industrial Electronics

Food, Beverages & Tobacco Furniture

Price range

Reach people who've shown interest in products within a specific price range.

☒ Low price range ☒ Mid-price range ☒ High price range

Target specific buyers

Target prospective buyers or activate the toggle to define who to include in your audience.

Include in your audience Yes ☒



Include people who visited between

To be, or not to be (only in the state)

Parfois insuffisant !




Options de **filtrage**/tri des données:

- URL (partage, favori, rafraichissement)
- Local/Session storage (persistence)

Nov 7, 2021 - Nov 13, 2021 (GMT)  

Ads [Create ad set](#)

[Review bid amount](#)

Delivery status	Budget left	Bid amount	Amount spent	Result	Cost per result	Ads
Not delivering	€308.00 €44.00 Daily 100%	Control by budget Conversions	-	-	-	2 Ads 
Not delivering	€308.00 €44.00 Daily 100%	Control by budget Conversions	-	-	-	2 Ads 
Ended	€10,000.00 €10,000.00 Lifetime 100%	Control by budget Conversions	-	-	-	2 Ads 

Optimiser le rendu DOM avec ngFor

TS app.component.ts X

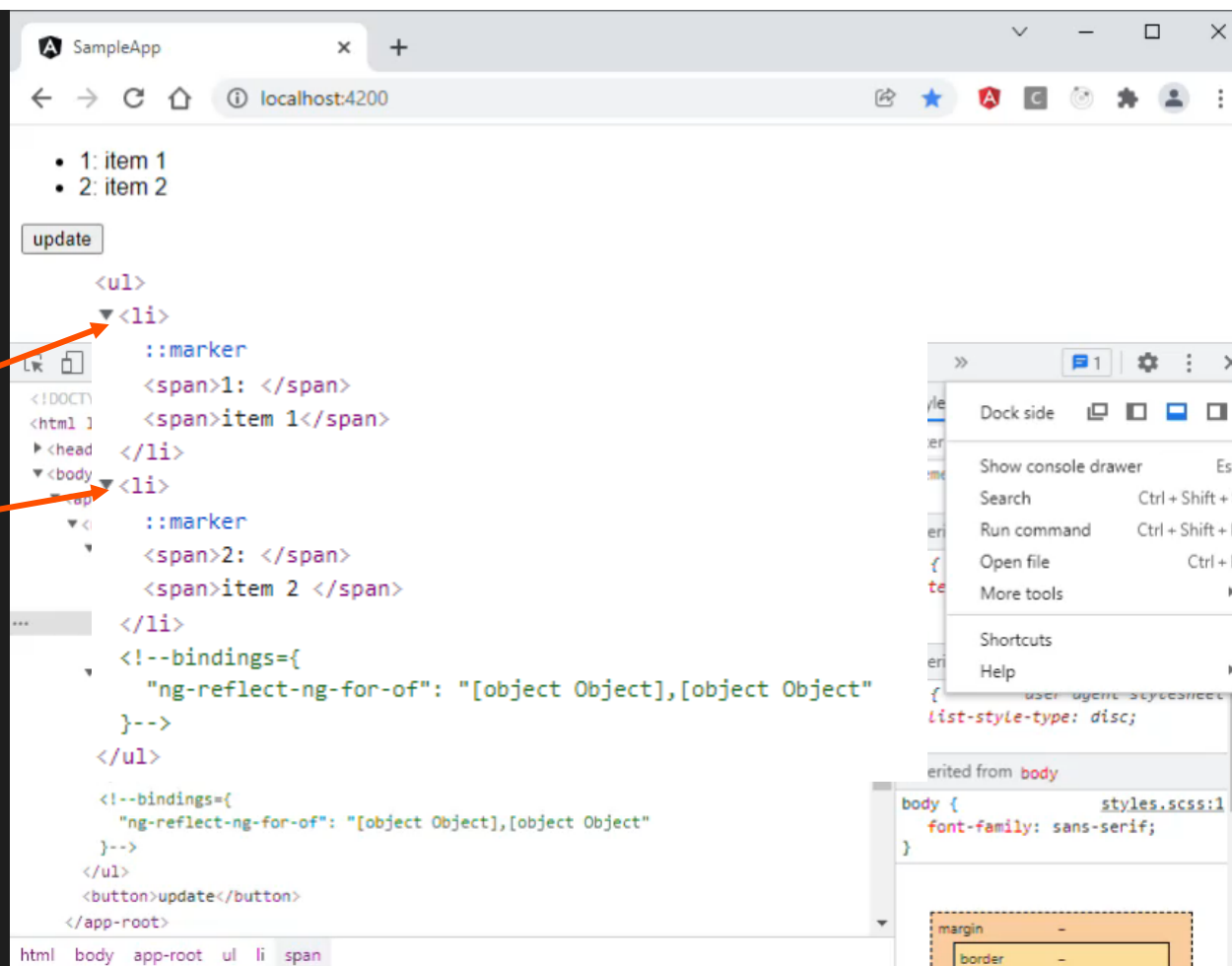
```

3  @Component({
4    selector: 'app-root',
5    template: `<ul>
6      <li *ngFor="let item of items">
7        <span>{{ item.id }}: </span>
8        <span>{{ item.name }}</span>
9      </li>
10    </ul>
11
12    <button (click)="update()">update</button>`,
13  })
14  export class AppComponent {
15    items = [
16      { id: 1, name: 'item 1' },
17      { id: 2, name: 'item 2 ' },
18    ];
19
20    update() {
21      this.items = [
22        { id: 1, name: 'item 1' },
23        { id: 2, name: this.items[1].name + '.' },
24      ];
25    }
26  }
27

```

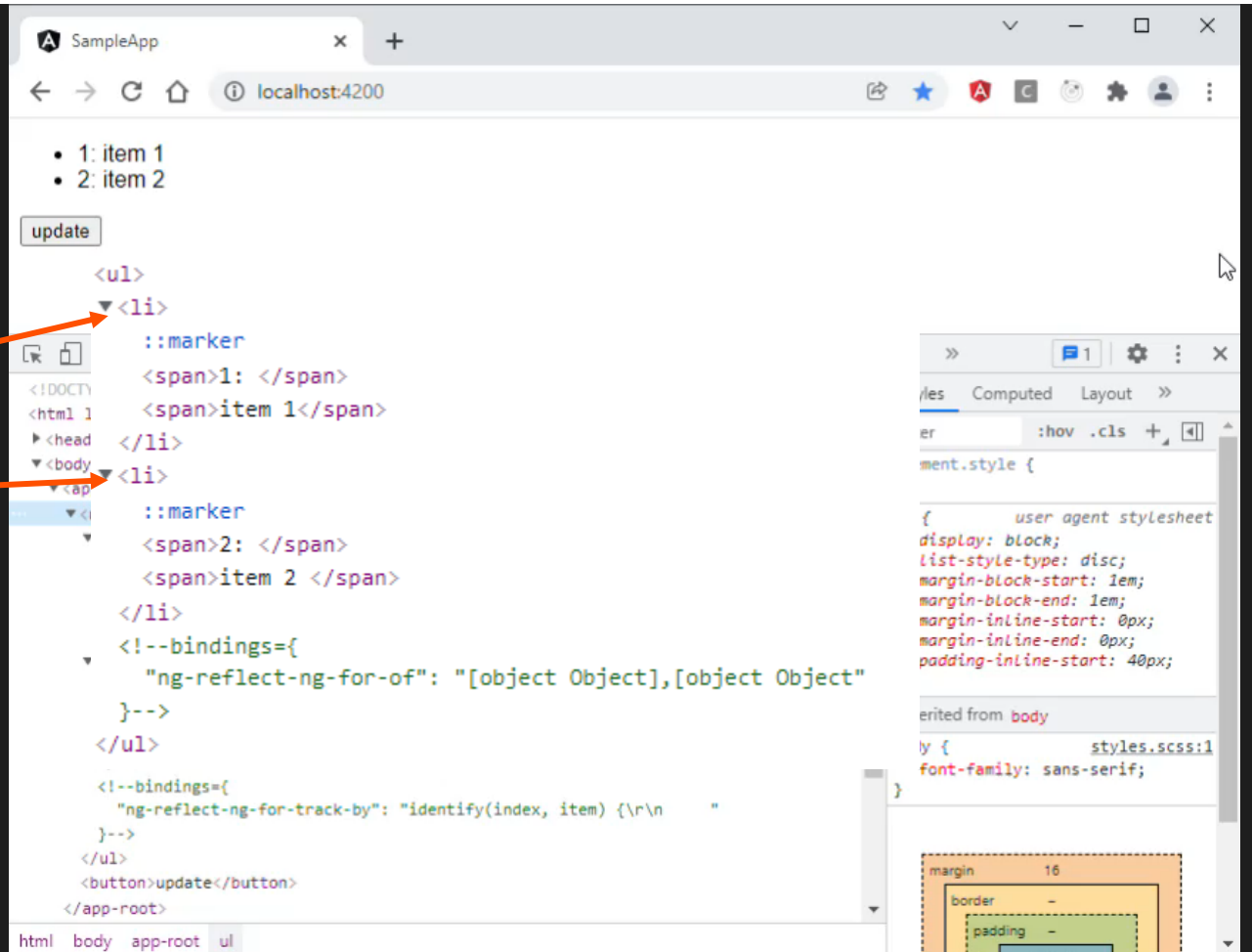
ref0

refl



Optimiser le rendu DOM avec ngFor

```
TS app.component.ts X
3 @Component({
4   selector: 'app-root',
5   template: `<ul>
6     <li *ngFor="let item of items; trackBy: identify">
7       <span>{{ item.id }}</span>
8       <span>{{ item.name }}</span>
9     </li>
10  </ul>
11
12  <button (click)="update()">update</button>`,
13 })
14 export class AppComponent {
15   items = [
16     { id: 1, name: 'item 1' },
17     { id: 2, name: 'item 2' },
18   ];
19
20   update() {
21     this.items = [
22       { id: 1, name: 'item 1' },
23       { id: 2, name: this.items[1].name + '.' },
24     ];
25   }
26
27   identify(index: number, item: { id: number; name: string }) {
28     return item.id;
29   }
30 }
31
```



Optimiser le rendu DOM avec ngFor

TS app.component.ts X

```
3  @Component({
4    selector: 'app-root',
5    template: `<ul>
6      <li *ngFor="let item of items; trackBy: 'id' | trackByFn">
7        <span>{{ item.id }}: </span>
8        <span>{{ item.name }}</span>
9      </li>
10     </ul>
11
12     <button (click)="update()">update</button>`,
13  })
14  export class AppComponent {
15    items = [
16      { id: 1, name: 'item 1' },
17      { id: 2, name: 'item 2 ' },
18    ];
19
20    update() {
21      this.items = [
22        { id: 1, name: 'item 1' },
23        { id: 2, name: this.items[1].name + '.' },
24      ];
25    }
26  }
27  |
```

TS track-by-fn.pipe.ts X

```
1  import { Pipe, PipeTransform } from '@angular/core';
2
3  @Pipe({ name: 'trackByFn' })
4  export class TrackByFnPipe implements PipeTransform {
5    transform<T>(key: keyof T): (_, v: T) => T[keyof T] {
6      return (_, v: T) => v[key];
7    }
8  }
9  |
```

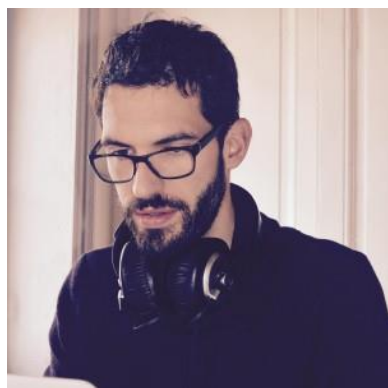
En résumé

- Un state par **type d'entité**
- Le structure du state doit matcher celle de l'API (normalisée)
- Déclarer des **sélecteurs unitaires**
- Utiliser la **composition**
- Ne pas **tout** mettre dans le state
- Ne pas mettre **que** dans le state
- Initialiser le state dans les TU
- Dispatcher les actions dans les tests UI
- Utiliser **trackBy**



Pour aller plus loin...

- `ChangeDetectionStrategy.OnPush`
- Utiliser des **pipes** (pures)
- Ne pas binder le résultat d'une fonction (coûteuse)
`{{ expensiveCall() }}`
- ...



ninja-squad.com

Cédric Exbrayat



Deviens un ninja avec Vue 3

Vue.js est le petit framework qui monte, et vient défier React et Angular. Cet ebook t'aide à en comprendre la philosophie, en se concentrant sur la version 3 et les suivantes, les nouveaux outils (comme ES2015, TypeScript, Vue CLI...), les librairies de l'écosystème (vue-router, @vue/test-utils, axios...), et chaque brique du framework de façon pragmatique. Dès la fin de la lecture, tu seras capable de démarrer ton projet et développer l'application de tes rêves ! Avec le **Pack Pro**, tu pourras télécharger un squelette d'application, plein de tests automatisés fournis. Tu pourras alors coder dans l'instant, étape par étape, pour construire une véritable application, en suivant ce que tu as appris dans le livre.

Ce livre a déjà été acheté par **755 personnes consentantes**, qui ont choisi de reverser **12% pour sponsoriser l'équipe qui développe le framework**. Le pack pro a déjà permis à **168 professionnels efficaces** d'apprendre Vue.js. Nos [chiffres de ventes](#) sont publics.

Il est mis à jour très régulièrement (et ces mises à jour sont gratuites pour tout acheteur !).

EN SAVOIR PLUS

ACHETER



Deviens un ninja avec Angular

Angular est une refonte complète du célèbre framework AngularJS. Cet ebook t'aide à en comprendre la philosophie (de Angular 2 à Angular 13.0), les nouveaux outils (comme ES2015, TypeScript, Webpack, Angular CLI...) et chaque brique du framework de façon pragmatique.

Dès la fin de la lecture, tu seras capable de démarrer ton projet et développer l'application de tes rêves ! Avec le **Pack Pro**, tu pourras télécharger un squelette d'application, plein de tests unitaires fournis. Tu pourras alors coder dans l'instant, étape par étape, pour construire une véritable application, en suivant ce que tu as appris dans le livre.

Ce livre a déjà été acheté par **6466 personnes consentantes**, qui ont choisi de reverser **10% à l'EFF**. Le pack pro a déjà permis à **1724 professionnels efficaces** d'apprendre Angular. Nos [chiffres de ventes](#) sont publics.

Il est mis à jour très régulièrement (et ces mises à jour sont gratuites pour tout acheteur !).

EN SAVOIR PLUS

ACHETER

Merci !

tinyurl.com/state-mgmt-ngxs

contact@xavierdupessey.com



CRITEO