7. RxJs / Promises

Introduction

Les observables et promise facilitent la gestion des évènements asynchrones. Comme vu dans le précédent chapitre, ils permettent par exemple de gérer le traitement de requêtes HTTP asynchrones sur un backend, sur le chargement de fichier, sur l'upload d'un fichier, etc.

RxJs

RxJs est une librairie en dehors de Angular, exploitée nativement ou au travers de nombreux framework. Angular se base sur l'Observable RxJs pour gérer les traitements asynchrones.

```
this.httpClient
  .get<any>("mes-assets/mon-fichier.json")
  .subscribe((result: any) => {
    // result contient le flux JSON
  });
```

Description complète de la lirbairie RxJs

Documentation Angular
Documentation officielle

Promesses

Les promesses, commes les observables, permettent la gestion des évènements asynchrones (comme vous avez pu le constater dans l'excercice précédent).

De la même façon, il est possible de les exploiter avec Angular pour gérer les traitements asynchrones.

Dans cette exemple, l'Observable renvoyé par la méthode `get` est transformé en Promise avec la métyhode `toPromise()`, permettant de récupérer le résultat de manière asynchrone avec les méthodes associées `then` et `catch`. Vous noterez l'exploitation des fonctions fléchées pour conserver le scope.

```
this.httpClient
  .get<any>("mes-assets/mon-fichier.json")
  .toPromise()
  .then((result: any) => {
    // result contient le flux JSON
  })
```

```
  .catch((error) => {
    // une erreur s'est produite
  });
```

Un chapitre sur les promesses a été dédié dans la formation Web de base / Chapitre ES6.

Différences entre RxJs Observables et Promise

- Un observable est 'lazy'. Tant que vous n'appelez pas subscribe() sur un observable que vous avez créé, le code que contient cet observable ne s'exécute pas. Alors qu'une promise, dès qu'elle est créée, est exécutée quoi qu'il arrive.
- Il est possible d'annuler un observable. Un observable retourne une fonction qui comporte tout le code de nettoyage à exécuter si vous avez décidé de l'annuler. Une promise en revanche, une fois qu'elle est créée, ira jusqu'au bout sans possibilité de renoncer à son exécution une fois qu'elle a commencé à s'exécuter. C'est un peu étrange dans le cas des promises utilisées pour faire une requête AJAX puisque l'objet XMLHttpRequest sous-jacent possède une méthode abort(), mais c'est ainsi à ce jour, on ne peut pas appeler abort() quand on utilise une promise (un projet de "cancellable promise" est ceci dit en projet pour ES7 ou ES8)
- Un observable peut être répété. S'il a précédemmement échoué (en raison d'une erreur réseau par exemple), il peut réussir lors d'une nouvelle exécution future (à l'aide de retry()) plus tard. Tandis qu'une promise conserve en cache le résultat de sa première exécution.

Exercice

Présentation

Application Météo.

L'objectif de cet excercice est de charger la météo au travers du module Angular HTTP, et d'exploiter RxJs.

Etapes d'utilisation de RxJS

1. Créer une nouvelle méthode dans le service `meteo` ayant la signature suivante :

```
public searchMeteoViaSubscribe(value: string): Observable<MeteoResult>
```

2. En lieu et place de la promesse, faites un appel simple à la méthode `get` de `http` .
3. Dans `meteo-view.component` appeler la méthode `searchMeteoViaSubscribe` , et utiliser la méthode `subscribe` pour souscrire à l'appel et récupérer le résultat.
4. Tester l'appel et valider que le resultat est fonctionnel.

Bravo, vous exploitez le gestionnaire de requête HTTP préconnisé par Angular, et vous avez pu tester la récupération de données asynchrone via RxJs ! Il n'y pas de bonne pratique sur l'utilisation de promesse ou de souscription, cela dépend des vos besoins (cf différences).

Nous allons maintenant revenir sur notre résultats de météo, et afficher la liste via les Directives.

Correction disponible ici