Marquer comme terminé

title: React Native - Séance 5

author: Cours de François Rioult francois.rioult@unicaen.fr

Résumé de la séance précédente

Nous avons découvert\ :

- comment concevoir une navigation avec un menu
- comment proposer une navigation optionnelle, par exemple selon que l'utilisateur est connecté ou non
- les jetons JWT d'authentification pour une API. Ces jeton contiennent l'identifiant de l'utilisateur et son signé par le serveur\ : ils sont infalsifiables
- nous avons réalisé une première brique de l'API\ : la connexion qui renvoie un jeton. Cette fonction est une *promesse*, réalisée en asynchrone et permet de traiter les erreurs.

Conception générale d'une application React

*Au fil de cette présentation, nous considérerons la réalisation d'une application de gestion de *TodoList*, en liaison avec une API CRUD appelée par des requêtes GraphQL.*

Lorsqu'on conçoit une application complète, il convient d'organiser les différents flux d'informations entre les différents composants\ :

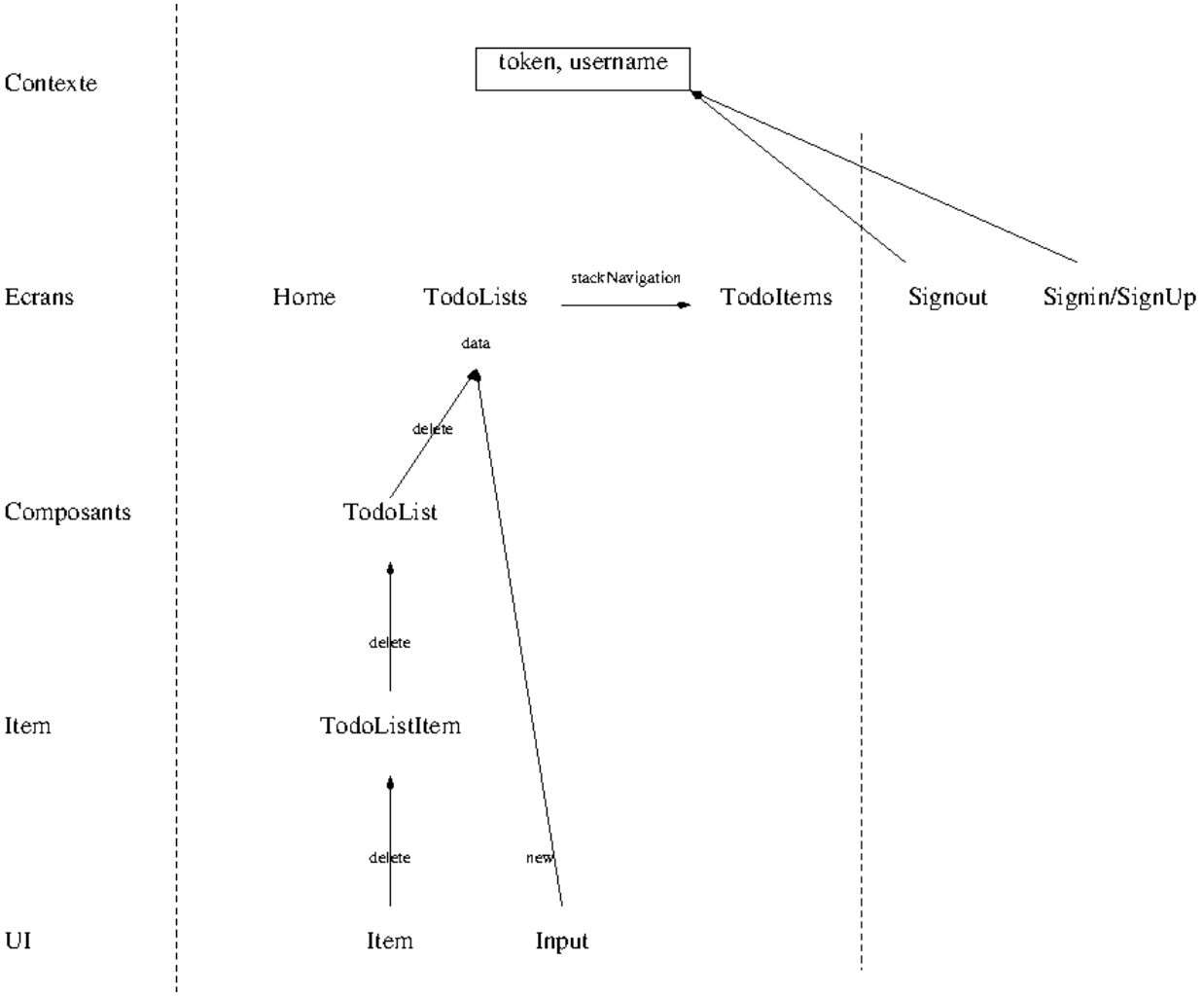
1. la première chose à décider concerne la réalisation du contexte, qui contient les variables globales, nécessaires pour la majorité des écrans, indépendamment de leur position dans la hiérarchie des composants. Ici, notre contexte contiendra le jeton et le nom de l'utilisateur.

2. ensuite, il faut déterminer les variables d'état et les composants qui accueilleront leur définition. Ces variables d'état (et leurs accesseurs) pourront potentiellement être transférées à des composants enfants par l'intermédiaire des *props*. Elles doivent donc être définies dans les composants le haut dans l'arbre des composants, en général dans les écrans.

3. déterminer quels composants doivent être des fonctions ou des classes. L'intérêt des classes est d'avoir un état interne, un cycle de vie du composant et des méthodes auxiliaires qui permettent de gérer le composant. En général, l'utilisation des *hook* d'état, de contexte et d'effet rendent inutile l'utilisation d'une classe.

Pour cette conception, on pourra s'organiser les idées en représentant la hiérarchie des composants.

?



Hook d'effet

Le *hook* d'effet `useEffect` est appelé par React à chaque mise à jour du DOM. Il permet d'éviter d'écrire les composants comme des classes, qui possèdent des méthodes de gestion du cycle de vie comme `componentDidMount`.

`useEffect` permet de mettre à jour l'état du composant en cas d'*effet de bord*, par exemple un appel initial à l'API pour charger la liste des TodoList. `useEffect` peut être pourvu d'un tableau de variables d'état. Il ne sera exécuté qu'en cas de modification de ces variables.

```
useEffect(() => {
    if (data.length == 0) {
      console.log('chargement initial')
      ...
    }
  }, [data])
```

Il est possible d'avoir plusieurs hooks `useEffect` pour gérer différents effets.

Modifié le: mercredi 2 octobre 2024, 17:00

◄ Bréviaire navigation / contexte / API

Choisir un élément

Aller à...

TP 5 ►

?

?