React 04

ESGI - 3ème année IW 2020

Introduction

React est une librairie centrée sur l’expérience développeur.

Le code devrait être modulable, divisé en composants idéalement réutilisables.

Différents modèle de conception (design patterns) et outils existent pour rendre le code plus composables.

Props.Children

La props children d’un composant React possède une description des composants entre ces balises

https://fr.reactjs.org/docs/composition-vs-inheritance.html

Inconvénient:
Pas de modification du contenu des enfants, on ne devrait pas connaître le contenu de children

Render props

Composant qui attend comme props une fonction de rendu:
https://fr.reactjs.org/docs/render-props.html

Avantages:
Permet de séparer la logique du rendu d'une fonctionnalité, comme par exemple: récupération des données d'une API / l'affichage de données

Inconvénient:
Il faut bien spécifier qu'on attend une fonction renvoyant un composant plutôt qu'un composant, que ce soit dans la documentation, dans les PropTypes ou encore une interface en Typescript

cloneElement

Fonction qui permet de cloner un élément React avec un objet Props en paramètre pour remplacer ou compléter ceux de l'élément cloné
https://reactjs.org/docs/react-api.html#cloneelement

Inconvénient:
Peut augmenter l'utilisation de la mémoire puisque le clone se fait à partir d'un élément déjà instancié

High Order Components

Permet de réutiliser la logique chez plusieurs composants, en déclarant explicitement une state, l'idée étant d'écrire une fonction qui prend un composant en paramètre et en renvoie un autre.
https://fr.reactjs.org/docs/higher-order-components.html

Inconvénient:
Risque de "Callback hell" à force d'imbriquer des HOCs

Compounds components

Compound components

Plusieurs composants qui partagent un état implicite

Utilisation de l’API Context pour partager cet état plutôt que de passer par les props

Avantage: permet de partager cet état avec une famille de composant plutôt que juste d’un parent à un enfant

Compound components

Cela est comparable aux balises <select /> et <option /> en HTML

Seules, ces balises n’ont pas d’utilité, mais en les combinant on obtient un sélecteur dont la donnée sélectionnée par l’utilisateur est partagée

On peut reproduire ce comportement avec les compound components, on aura alors:

- Un composant fournissant la state partagée: le Provider
- Un context accessible aux enfants et sous-enfants qui peuvent se brancher dessus pour lire et interagir avec cette state

Mise en place

- Création et exportation du Context et de son Provider avec les fonctions de l'API, par exemple dans un fichier mycontext.js :

```
export const MyContext = createContext(initData = {});
export const MyContextProvider = MyContext.Provider;
```

- Écriture d'un composant parent utilisant comme balises le provider:

```
<MyContextProvider value={mySharedData}></MyContextProvider>
```

- Utilisation de MyContext dans les composants enfants:

```
const context = useContext(MyContext);
const { mySharedData } = context;
```

Demo

TP

Réécrire en partie votre TP rendu pour utiliser les compounds components et ainsi éviter le problème de passage de données par les props