React 02

ESGI - 3ème année IW 2020

Cycle de vie d’un composant

Présentation

Lorsqu’on accède à une page qui affiche un composant React:

- Une phase de **montage** (mounting), première exécution du code du composant
- Une ou plusieurs phase de **mise à jour** (updating), lorsqu’une propriété (props) passée au composant change, si son état (state) change, ou encore si le parent du composant se met à jour
- Une phase de **démontage** (unmounting), exécuté lorsque le composant est retiré de l’affichage (changement de page etc)

React moderne: les hooks

- Rétrocompatible avec les classes
- Paradigme fonctionnel
- API de React depuis la version 16.8
- Code plus concis et plus facile à factoriser ainsi qu’à partager entre les composants
- Un hook = une fonction dont le nom commence par “use”

Utilisation des hooks

Diagramme explicatif: React lifecycle Hooks diagram

Documentation: Référence de l'API des Hooks – React

Les deux principaux hooks

useState

Permet de créer une variable dans le state d'un composant
Exemple:

```
const [state, setState] = useState(initialState);
```

- useState renvoie un array contenant deux choses:
  - Une référence à une donnée dans la state du composant
  - Une fonction pour mettre à jour sa valeur
- Il prend en paramètre une valeur d'initialisation (un nombre, une chaîne de caractères etc)
- Pas le même comportement que this.setState: pas de merge automatique

useState: démo

useEffect

Permet d’exécuter une fonction après la phase de rendu et de la re-exécuter si une dépendance change

ex: `useEffect(maFonction, [dependanceA, dependanceB])`

- Sert pour les mutations, les timers, les souscriptions à une API etc
- Sert à éviter de bloquer la phase de rendu pour attendre des données indisponible instantanément
- Peut renvoyer une fonction de nettoyage exécutée avant une nouvelle exécution ou/et au démontage du composant
- Permet de se brancher au cycle de vie du composant

useEffect et cycle de vie

Si aucun paramètre de dépendance n’est passé, la callback est exécuté à chaque rendu (=> componentDidUpdate)

Si un tableau vide ([ ]) est passé en dépendance, la callback ne sera appelé qu’au premier rendu (=> componentDidMount)

Si un tableau vide ([ ]) est passé en dépendance mais que la callback ne fait que renvoyer une fonction de nettoyage, elle ne sera exécuté qu’au démontage du composant (=> componentDidUnmount)

useEffect: démo

Les autres hooks

- Hooks situationnels
- Utilisation rare
- Permettent différents niveaux d’optimisation

On peut aussi écrire nos hooks personnalisés, utile pour factoriser et réutiliser le hook dans plusieurs composants, mais bien respecter les règles des hooks

TP

Choisir une API pour récupérer des données distantes (https://github.com/public-apis/public-apis)

Écrire plusieurs composants pour:

- Afficher un ensemble de données de l'API
- Filtrer selon plusieurs critères en faisant un nouvel appel à l'API
- Sélectionner un ou plusieurs éléments affichés pour les mettre en surbrillance (changer leur cadre, leur couleur, …)
- Un bouton submit pour transmettre ces données à un autre composant qui devra les afficher différemment