Débutez avec React

8 heures Moyenne

Mis à jour le 07/03/2022

Stockez et récupérez des données avec le state et les effets

Vous n'avez pas validé ce quiz.

Vous n'avez pas atteint le seuil de validation de cet exercice, c'est-à-dire 70%. Ce n'est pas très grave car vous pourrez repasser le quiz dans 24h.

Compétences évaluées

✗ Stocker et récupérer des données avec le state et les effets

Question 1

Identifiez la phrase incorrecte :

- ○ Le state local existe à l'intérieur d'un composant
- ✗ ◉ Ce composant peut être re-render autant de fois que l'on veut, mais la data sera préservée
- ○ On utilise le state local dans un composant fonction avec `useState`
- ✓ ○ On met à jour notre state en faisant `useState(nouvelleValeur)`

Toutes ces affirmations sont exactes... sauf la dernière. La valeur entre parenthèses de useState correspond à la valeur initiale de notre state. Souvenez-vous la syntaxe est `const [variableDEtat, fonctionDeMiseAJour] = useState(valeurInitiale)` *. L'intrus était donc la dernière option : réponse 4.*

Question 2

Vous voulez créer un state local `cartValue` initialisé à 0 qui vous permet de sauvegarder le prix des articles que vous ajoutez à votre panier.

Quelle(s) syntaxe(s) vous permet(tent) de récupérer et updater votre state :

Attention, plusieurs réponses sont possibles.

- [ ]
```javascript
const updateCartValue = useState({ cartValue: 0 })
```

- [x]
```javascript
const cartState = useState(0)
const cartValue = cartState[0]
const updateCartValue = cartState[1]
```

- [ ]
```javascript
const [cartValue, updateCartValue] = useState(0)
```

- [x]
```javascript
const [cartValue, updateCartValue] = useState(1)
```

La syntaxe de la première proposition ne correspond pas du tout à la manière d'utiliser `useState` .

La syntaxe de la dernière option est correcte. En revanche, la valeur initiale est de 1, alors qu'on voulait une valeur initiale de 0.

La deuxième proposition fonctionne bien ainsi que la 3, même si la meilleure pratique reste la 3.

Question 3

Quelles affirmations sont vraies ?

Attention, plusieurs réponses sont possibles.

- [ ] useEffect et useState doivent être importés depuis le package "react-hooks"
- [ ] useEffect et useState sont les deux seuls hooks existants
- [x] Les hooks n'existaient pas aux débuts de React
- [x] Un hook est une fonction qui permet de « se brancher » (*to hook up*) sur des fonctionnalités React

`useEffect` et `useState` s'importent directement depuis React en faisant

```javascript
import { useState, useEffect } from 'react'
```

Par ailleurs, il existe de nombreux autres hooks (useRef, useLayoutEffect, etc.).

En revanche, les affirmations 3 et 4 sont exactes. Les hooks sont arrivés avec React 16.8 et permettent de se brancher sur des fonctionnalités React. Il fallait donc répondre 3 et 4.

Question 4

Vous voulez partager une variable d'état entre plusieurs composants. Pour ça, vous devrez :

Attention, plusieurs réponses sont possibles.

- ✗ ☐ Faire remonter ces données vers le state local du plus proche composant qui est un ancêtre commun
- ✓ ☑ Faire redescendre ces infos avec des *props* jusqu'aux composants qui en ont besoin
- ✓ ☑ Faire « remonter » les demandes d'update en utilisant la fonction de mise à jour du state.
- ☐ Déclarer plusieurs variables de *state*

Ici, vous voulez partager une seule variable d'état entre plusieurs composants.

Il faut donc faire remonter cette variable vers le plus proche parent commun, faire redescendre les données avec les props, et faire remonter les demandes d'update en passant la fonction d'update dans les props.

En revanche, déclarer plusieurs variables de state ne m'aidera pas ici.

Les bonnes réponses sont donc la 1, la 2 et la 3.

Question 5

Quelles règles concernant `useEffect` sont exactes ?

Attention, plusieurs réponses sont possibles.

- ✓ ☑ `useEffect` doit toujours être appelé depuis un composant fonction
- ☐ Le state ne peut pas être utilisé dans `useEffect`
- ✗ ☑ Il n'est pas possible d'avoir plusieurs `useEffect` dans un même composant
- ✓ ☑ Il faut toujours appeler `useEffect` au niveau racine votre composant, sans l'inclure dans une condition, une fonction imbriquée ou une boucle

Comme les autres hooks, useEffect doit forcément être appelé depuis un composant fonction. Par ailleurs, il est important de ne pas appeler vos hooks à l'intérieur d'une condition : ils doivent être à la racine de votre composant.
En revanche, pas de soucis pour utiliser plusieurs `useEffect` dans un même composant ou pour utiliser le state dans `useEffect`. Les réponses 2 et 3 sont fausses.

Les bonnes réponses sont donc la 1 et la 4 !

Question 6

Vous créez un composant `Welcome`.

javascript

```
1 function Welcome() {
2     return <div>Welcome</div>
```

```
3 }
```

Vous souhaitez également y ajouter une alerte, qui s'affiche uniquement au premier chargement de la page.

Où est-ce que vous devez écrire votre alerte ?

○ javascript

```javascript
function Welcome() {
    alert('Welcome! ✳')
    return <div>Welcome</div>
}
```

○ javascript

```javascript
function Welcome() {
    if (!localStorage.getItem('welcome')) {
        useEffect(() => {
            alert('Welcome! ✳')
        })
    }
    return <div>Welcome</div>
}
```

✓ ○ javascript

```javascript
function Welcome() {
    useEffect(() => {
        alert('Welcome! ✳')
    }, [])
    return <div>Welcome</div>
}
```

✗ ◉ javascript

```javascript
function Welcome() {
    useEffect(() => {
        alert('Welcome! ✳')
    }, [Welcome])
    return <div>Welcome</div>
}
```

Si vous souhaitez déclencher un effet uniquement après le premier rendu de votre composant, il suffit d'utiliser le tableau de dépendances... vide !
La bonne réponse était donc la 3 !

Question 7

Vous avez un composant `Cart` .

javascript

```javascript
function Cart() {
    const [cart, updateCart] = useState(0)
    return <div>
        <h2>Mon panier</h2>
        {cart}
        <button onClick={() => updateCart(cart+1)}>Ajouter un élément au panier</button>
    </div>
```

```
8 }
```

Si vous voulez déclencher une alerte qui s'affiche uniquement après la mise à jour de votre panier, quel snippet de code fonctionne ?

- [ ]

javascript

```javascript
function Cart() {
    const [cart, updateCart] = useState(0)
    alert(`Mon panier contient ${cart} éléments`)
    return <div>
        <h2>Mon panier</h2>
        {cart}
        <button onClick={() => updateCart(cart+1)}>Ajouter un élément au panier</button>
    </div>
}
```

- [ ]

javascript

```javascript
function Cart() {
    const [cart, updateCart] = useState(0)
    return (<div>
        <h2>Mon panier</h2>
        {cart}
        <button onClick={() => updateCart(cart+1)}>Ajouter un élément au panier</button>
    </div>)
    alert(`Mon panier contient ${cart} éléments`)
}
```

- [ ]

javascript

```javascript
function Cart() {
    const [cart, updateCart] = useState(0)
    useEffect(() => {
       alert(`Mon panier contient ${cart} éléments`)
    })
    return (<div>
        <h2>Mon panier</h2>
        {cart}
        <button onClick={() => updateCart(cart+1)}>Ajouter un élément au panier</button>
    </div>)
}
```

- [x]

javascript

```javascript
function Cart() {
    const [cart, updateCart] = useState(0)
    useEffect(() => {
       alert(`Mon panier contient ${cart} éléments`)
    }, [cart])
    return (<div>
        <h2>Mon panier</h2>
        {cart}
        <button onClick={() => updateCart(cart+1)}>Ajouter un élément au panier</button>
    </div>)
}
```

Ici, vous pouvez passer par `useEffect` . Les deux premières réponses ne l'utilisent pas. Si vous testez le code de la réponse 1, vous verrez que l'alerte s'affiche de manière intempestive : dès que vous ouvrez votre fenêtre, quand vous voulez la fermer. Cette option ne fonctionne pas. La réponse 2 ne fonctionne pas non plus car, située après le return, l'alerte est donc située dans du code inatteignable.

La réponse 3 a bien `useEffect`, mais pour que l'action n'ait lieu que lorsque vous mettez votre panier à jour, vous devez utiliser le tableau de dépendances. La bonne réponse est donc la réponse 4 !

Question 8

Vous êtes dans le composant parent le plus élevé dans mon arborescence de composants, `App.js`, où vous avez le code suivant :

javascript

```javascript
function App() {
    return <div>
        <Header />
        <Cart />
        <Footer />
    </div>
}
```

Et votre Cart correspond au code suivant :

javascript

```javascript
function Cart() {
    const [cartValue, updateCartValue] = useState(0)
    return <div>
        <button onClick={() => updateCartValue(cartValue + 1)}>Ajoutez-moi</button>
        <h3>Total de mon panier : {cartValue}</h3>
    </div>
}
```

Vous vous rendez compte que votre Header doit afficher le contenu de votre panier, et votre Header doit changer de couleur en fonction de votre panier : il devient blanc lorsque votre panier est vide et vert lorsqu'il est plein.

Quelle est la meilleure manière d'écrire votre fichier `App.js` si vous voulez pouvoir faire tout ça ?

- ○

javascript

```javascript
function App({cartValue, updateCartValue}) {
    return <div>
        <Header cartValue={cartValue} />
        <Cart updateCartValue={updateCartValue} />
        <Footer cartValue={cartValue} />
    </div>
}
```

- ○

javascript

```javascript
function App() {
    const [cartValue, updateCartValue] = useState(0)
    return <div>
        <Header cartValue={cartValue} />
        <Cart cartValue={cartValue} onClick={() => updateCartValue(cartValue + 1) } />
        <Footer cartValue={cartValue} />
    </div>
}
```

- ○

javascript

```javascript
function App() {
```

```
    const [cartValue, updateCartValue] = useState(0)
    return <div>
        <Header  />
        <Cart />
        <Footer />
    </div>
}
```

javascript

```
function App() {
    const [cartValue, updateCartValue] = useState(0)
    return <div>
        <Header cartValue={cartValue} />
        <Cart cartValue={cartValue} updateCartValue={updateCartValue} />
        <Footer cartValue={cartValue} />
    </div>
}
```

La première option récupère cartValue et updateCartValue dans ses props. Pour ça, il faudrait qu'ils soient passés par un composant parent. Or, comme précisé dans la consigne, il s'agit du composant le plus haut dans l'arborescence des composants. La première option ne marche donc pas.

Dans sa logique, la deuxième solution pourrait paraître tentante. Sauf qu'on ne peut pas accéder à onClick directement sur un composant : il faut préciser cet événement sur un élément du DOM. Ici, il s'agit juste d'une prop qui s'appelle onClick. En plus, on ne lui précise pas la valeur du panier, ce qui ne nous permet pas d'afficher le total.

La troisième option quant à elle ne précise aucune prop : rien ne va !

La bonne réponse était donc la réponse 4 ✨

DÉCLENCHEZ DES EFFETS AVEC USEEFFECT

REVENEZ SUR VOS ACQUIS

Le professeur

Alexia Toulmet

Développeuse Fullstack React/Node freelance passionnée par le frontend 👩‍💻

POUR LES ENTREPRISES

EN PLUS

Français