Marquer comme terminé

Résumé de la Séance 1

- React Native utilise la librairie React et utilise les composants natifs du périphérique.
- `expo-cli` permet d'initialiser et de construire une application React Native, qui pourra être exécutée dans un navigateur ou sur un périphérique Android ou iOS. Le site https://expo.dev permet de partager des applications, fournit l'infrastructure pour les exécuter, les construire et les déposer sur les *stores*.
- pour exécuter une application React Native
  - sur le web\ : il faut utiliser un serveur Node.js
  - sur Android\ : il faut la construire à l'aide d'AndroidStudio
  - sur iOS : il faut la construire à l'aide d'XCode (disponible uniquement sur un ordinateur Apple)
- React Native permet de définir ses propres composants à partir des composants natifs.
- un composant personnalisé est une fonction qui renvoie du JSX, un mélange de Javascript et de XML.
- les composants fonctions peuvent utiliser le hook `useState` pour définir un état et le modifier.
- les composants pères peuvent utiliser des composants fils en leur fournissant des *props*, accessibles uniquement en lecture par le fils.

Communications fils -> père

Les communications père -> fils sont gérées par les props, ce sont des paramètres passés au fils au moment de sa création. Lorsqu'ils sont modifiés par le père, ils sont automatiquement mis à jour dans le fils.

Cependant, il est fréquent de vouloir modifier l'état du père depuis le fils. Par exemple, un bouton chez les fils modifie un compteur chez le père. Pour cela, on va utiliser les *props* pour que le père indique au fils quelle fonction appeler pour modifier son état.

Dans l'exemple ci-dessous, l'information est dynamiquement mise à jour, dans les deux sens\ : - père vers fils : la *props* `count` met à jour dynamiquement le composant Text du fils - fils vers père : l'appui sur le bouton appelle la *props* `onPressed`, qui met à jour l'état du père.

Dans le composant père, on définit une fonction qui sera appelée par le fils, à qui on la passe par les *props*\ :

```
export default function App() {
  const [count, setCount] = useState(0);

  const onPressed = () => {setCount(count + 1)}

  return (
    <View style={styles.container}>
      <Text>Père : le bouton a été pressé {count} fois</Text>
      <MonComposant onPressed={onPressed} count={count}/>
    </View>
  );
}
```

Dans le composant fils, on appelle cette fonction issue des props\ :

```
export default function MonComposant(props) {

    return (
        <>
            <Text>Fils : le bouton a été pressé {props.count} fois</Text>
            <Button title='Press me' onPress={props.onPressed} />
        </>
    )
}
```

Composant sous forme de classe

Les composants sous forme de classe sont obsolètes. Il faut désormais rédiger les composants sous forme de fonction, comme décrit lors de la première séance.

?

Cependant, il subsiste beaucoup de documentation sur le web qui utilise des composants sous la forme de classe. Ce document peut permettre de comprendre cette syntaxe.

Modifié le: lundi 9 septembre 2024, 11:25

◄ TP 1

Choisir un élément

Aller à...

TP 2 ►

?