Qu'est ce que react?

React est une bibliothèque JavaScript open source développée par Facebook depuis 2013. Elle est utilisée pour construire des interfaces utilisateur, et est principalement utilisée pour construire des applications web monopage. React peut également être utilisé pour construire des applications mobiles. Il est utilisé par Facebook, Instagram, Netflix, Airbnb, Pinterest, Walmart, Flipkart, et bien d'autres.

React permet de créer des composants réutilisables. Cela signifie que chaque élément de l'interface utilisateur est indépendant et peut être utilisé à plusieurs reprises. React fournit des composants de base et permet de créer des composants personnalisés.

React est une bibliothèque JavaScript, ce qui signifie qu'il n'est pas une application complète. Il est utilisé avec d'autres bibliothèques ou frameworks pour créer des applications complètes. Il est utilisé avec des bibliothèques comme Redux pour créer des applications monopages.

---

le DOM virtuel

Le DOM virtuel est un objet JavaScript qui est une représentation du DOM. Il est utilisé pour représenter l'interface utilisateur de l'application. Il est utilisé pour gérer les changements dans l'interface utilisateur. Il est utilisé pour mettre à jour l'interface utilisateur lorsque les données changent.

JSX

JSX est une extension de JavaScript. Il est utilisé avec React.

- Il est utilisé pour décrire à React comment l'interface utilisateur doit être affichée. - Il est utilisé pour décrire l'interface utilisateur en utilisant des balises HTML.
- Il est utilisé pour décrire l'interface utilisateur en utilisant des balises personnalisées.

Créer une application React

Pour créer une application React, vous devez utiliser la commande `create-react-app`. Cela crée un nouveau projet React avec tous les fichiers nécessaires.

```
npx create-react-app my-app
cd my-app
npm start
```

Composants

Les composants sont des éléments réutilisables de l'interface utilisateur. Ils sont utilisés pour créer des interfaces utilisateur réutilisables. Les composants peuvent être créés à l'aide de fonctions ou de classes.

Composants fonctionnels

Les composants fonctionnels sont des fonctions JavaScript qui retournent un élément React. Ils sont utilisés pour créer des composants simples qui ne contiennent pas d'état.

```
function Welcome() {
  return <h1>Hello world!!</h1>;
}
```

Le nom d'un composant fonctionnel doit commencer par une lettre majuscule.

On oublie pas d'exporter le composant pour pouvoir l'utiliser dans d'autres fichiers.

```
export default Welcome;
```

Importer un composant

Pour importer un composant, on utilise l'instruction `import`. On peut importer un composant depuis un fichier ou depuis une bibliothèque.

```
import Welcome from "./Welcome";
```

On appel le composant dans le fichier où on l'a importé.

```
<Welcome />
```

Les props

Les **props** sont des arguments passés à un composant. Ils sont utilisés pour passer des données à un composant. Les props sont passés à un composant en tant qu'attributs HTML.

```
function Welcome(props) {
  return <h1>Hello {props.name}!!</h1>;
}
```

Les proprieté sont passées du composant parent au composant enfant. On peut passer des props à un composant enfant en utilisant l'attribut `name`. Quand on appel le composant, on passe la valeur de l'attribut `name` entre les balises.

```
<Welcome name="John" />
```

On peut passer plusieurs **props** à un composant.

```
<Welcome name="John" age="25" />
```

On peut utiliser le destructuring pour récupérer les **props**.

```
function Welcome({ name, age }) {
  return <h1>Hello {name}!!</h1>;
}
```

Les états

Les états sont des données qui peuvent changer dans le temps. Ils sont utilisés pour stocker des données qui peuvent changer dans le temps. Les états sont utilisés pour créer des composants dynamiques.

Déclarer un état

Pour déclarer un état, on utilise l'instruction `useState`. On passe la valeur initiale de l'état en paramètre. On récupère la **valeur** de l'état et la **fonction** pour modifier l'état dans un tableau.

```
const [count, setCount] = useState(0);
```

On oublie pas d'importer `useState` depuis la bibliothèque `react`.

```
import { useState } from "react";
```

Modifier un état

Pour modifier un état, on utilise la fonction `setCount`. On passe la nouvelle valeur de l'état en paramètre.

```
setCount(1);
```

---

evenements

Les événements sont des actions qui sont déclenchées par l'utilisateur. Ils sont utilisés pour déclencher des fonctions lorsque l'utilisateur effectue une action.

Déclencher une fonction

Pour déclencher une fonction, on utilise l'attribut `onClick`. On passe la fonction à déclencher en paramètre.

```
<button onClick={increment}>+</button>
```

Les conditions

La condition ternaire est une condition qui retourne une valeur en fonction d'une condition. Elle est utilisée pour afficher des éléments conditionnellement.

```
{
  count === 0 ? <p>Zero</p> : <p>Not zero</p>;
}
```

Les listes

Les listes sont des tableaux de données. Elles sont utilisées pour afficher des données dans une liste.

Afficher une liste

Pour afficher une liste, on utilise la méthode `map`. On passe une fonction en paramètre. Cette fonction est appelée pour chaque élément du tableau. On retourne un élément React pour chaque élément du tableau.

```
{
  tags.map((tag) => <li key={tag}>{tag}</li>);
}
```

On utilise l'attribut `key` pour spécifier une clé unique pour chaque élément de la liste. On utilise l'attribut `key` pour améliorer les performances de React.

---

react-router-dom

React-router-dom est une bibliothèque qui permet de créer des applications web avec plusieurs pages. Elle est utilisée pour créer des applications web avec plusieurs pages.

Installer react-router-dom

Pour installer react-router-dom, on utilise la commande `npm`.

```
npm install react-router-dom
```

Importer react-router-dom

Pour importer react-router-dom, on utilise l'instruction `import`.

```
import { BrowserRouter } from "react-router-dom";
```

On englobe l'application dans le composant `BrowserRouter`.

```
<BrowserRouter>
  <App />
</BrowserRouter>
```

Créer des liens

Pour créer des liens, on utilise le composant `Link`. On passe l'attribut `to` pour spécifier l'URL du lien.

```
<Link to="/about">About</Link>
```

Créer des routes

Pour créer des routes, on utilise le composant `Route`. On passe l'attribut `path` pour spécifier l'URL de la route. On passe l'attribut `element` pour spécifier le composant à afficher.

```
<Route path="/about" element={<Composant />} />
```

on englobe les routes dans le composant `Routes`.

```
<Routes>
  <Route path="/about" element={<Composant />} />
</Routes>
```
