* React est une bibliothèque JavaScript développée par Facebook pour créer des interfaces utilisateur. Il est principalement utilisé pour construire des interfaces utilisateur dynamiques et interactives pour des applications web modernes.

Pour créer une app react : npx create-react-app nom-app

Cd nom-app

Npm start

Localhost :3000

Rfc react fucntion components

* **Philosophie de React :**
  + **Composants** : React se base sur une architecture de composants, où l'interface utilisateur est divisée en composants réutilisables.
  + **Unidirectional Data Flow** : Les données circulent dans une direction (du parent au composant enfant).
  + **Virtual DOM** : React utilise un DOM virtuel pour optimiser les mises à jour de l'interface utilisateur.

**2. Concepts de Base**

* **Composants :**
  + **Fonctionnels** : Définis en tant que fonctions JavaScript. Utilisés principalement pour des composants simples.
  + **Classe** : Définis en utilisant la syntaxe de classe ES6. Utilisés pour des composants plus complexes nécessitant un état interne.

jsx

Copier le code

// Composant Fonctionnel

function MyComponent() {

return <h1>Hello, World!</h1>;

}

// Composant de Classe

class MyComponent extends React.Component {

render() {

return <h1>Hello, World!</h1>;

}

}

* **JSX (JavaScript XML) :** Une syntaxe qui permet d'écrire des éléments HTML et JavaScript dans le même fichier. JSX est compilé en JavaScript.

jsx

Copier le code

const element = <h1>Hello, World!</h1>;

* **Props :** Les propriétés (props) sont des paramètres passés aux composants. Ils permettent de transmettre des données d'un composant parent à un composant enfant.

jsx

Copier le code

function Greeting(props) {

return <h1>Hello, {props.name}</h1>;

}

* **State :** L'état (state) est un objet qui détermine comment un composant se comporte et ce qu'il affiche. Il est mutable et géré au niveau du composant.

jsx

Copier le code

class Counter extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

render() {

return (

<div>

<p>You clicked {this.state.count} times</p>

<button onClick={() => this.setState({ count: this.state.count + 1 })}>

Click me

</button>

</div>

);

}

}

**3. Lifecycle Methods**

Les méthodes de cycle de vie permettent de contrôler le comportement des composants à différentes étapes de leur existence.

* **Mounting** : Lorsqu'un composant est ajouté au DOM (componentDidMount).
* **Updating** : Lorsqu'un composant est mis à jour (componentDidUpdate).
* **Unmounting** : Lorsqu'un composant est retiré du DOM (componentWillUnmount).

**4. Hooks**

Les hooks sont des fonctions qui permettent d'utiliser l'état et d'autres fonctionnalités de React dans les composants fonctionnels.

* **useState** : Permet de gérer l'état dans un composant fonctionnel.

jsx

Copier le code

import React, { useState } from 'react';

function Counter() {

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

return (

<div>

<p>You clicked {count} times</p>

<button onClick={() => setCount(count + 1)}>Click me</button>

</div>

);

}

* **useEffect** : Permet d'effectuer des effets de bord dans un composant fonctionnel (comme les appels API).

jsx

Copier le code

import React, { useState, useEffect } from 'react';

function DataFetcher() {

const [data, setData] = useState(null);

useEffect(() => {

fetch('https://api.example.com/data')

.then(response => response.json())

.then(data => setData(data));

}, []); // Le tableau vide signifie que l'effet se produit une seule fois après le premier rendu

return <div>{data ? <p>{data.title}</p> : <p>Loading...</p>}</div>;

}

**5. Routing**

Pour ajouter la navigation à votre application React, vous utilisez react-router-dom.

* **Installation :**

bash

Copier le code

npm install react-router-dom

* **Exemple d'utilisation :**

jsx

Copier le code

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import Home from './Home';

import About from './About';

function App() {

return (

<Router>

<Switch>

<Route path="/" exact component={Home} />

<Route path="/about" component={About} />

</Switch>

</Router>

);

}

**6. Gestion d'état**

Pour une gestion d'état plus complexe, vous pouvez utiliser des bibliothèques comme Redux ou le Context API.

* **Redux :** Une bibliothèque pour gérer l'état global de l'application de manière prévisible.

bash

Copier le code

npm install redux react-redux

* **Context API :** Un mécanisme intégré dans React pour partager des états entre composants sans avoir à passer les props manuellement.

jsx

Copier le code

import React, { createContext, useState } from 'react';

const MyContext = createContext();

function MyProvider({ children }) {

const [value, setValue] = useState('initial value');

return (

<MyContext.Provider value={{ value, setValue }}>

{children}

</MyContext.Provider>

);

}

function MyComponent() {

const { value, setValue } = React.useContext(MyContext);

return (

<div>

<p>{value}</p>

<button onClick={() => setValue('new value')}>Change Value</button>

</div>

);

}

**7. Styling**

* **CSS Modules :** Un moyen de styliser vos composants avec des classes locales pour éviter les conflits de noms.

jsx

Copier le code

import styles from './MyComponent.module.css';

function MyComponent() {

return <div className={styles.myClass}>Styled Component</div>;

}

* **Styled Components :** Une bibliothèque pour écrire du CSS dans vos fichiers JavaScript.

bash

Copier le code

npm install styled-components

jsx

Copier le code

import styled from 'styled-components';

const Button = styled.button`

background: blue;

color: white;

`;

function MyComponent() {

return <Button>Styled Button</Button>;

}

**8. Optimisation des Performances**

* **React.memo** : Un HOC qui permet d'optimiser les performances en évitant les rendus inutiles des composants fonctionnels.
* **useCallback et useMemo** : Hooks pour mémoriser les fonctions et les valeurs calculées afin d'éviter les recalculs inutiles.

**9. Outils et Environnement de Développement**

* **Create React App** : Un outil pour créer une application React avec une configuration par défaut.

bash

Copier le code

npx create-react-app my-app

* **React DevTools** : Une extension pour le navigateur qui permet de déboguer les applications React.
* **Jest** : Un framework de test pour écrire des tests pour vos composants React.

bash

Copier le code

npm install --save-dev jest

* **ESLint et Prettier** : Outils pour le linting et le formatage du code.

bash

Copier le code

npm install --save-dev eslint prettier eslint-plugin-react eslint-config-prettier

**10. Déploiement**

* **Vercel, Netlify, GitHub Pages** : Plateformes courantes pour déployer des applications React.
* **Docker** : Pour créer des conteneurs qui encapsulent votre application et ses dépendances.

En maîtrisant ces concepts et outils, vous serez bien équipé pour développer des applications React robustes et performantes.

4o mini