Explication simple de l'utilisation de `fetch` et `axios` dans React pour récupérer des données :

---

1. Utilisation de `fetch` :

`fetch` est une méthode native de JavaScript pour envoyer des requêtes HTTP et récupérer des données.

Exemple simplifié :

```
import React, { useEffect, useState } from "react";

const App = () => {
  const [data, setData] = useState([]); // Stocke les données
  const [loading, setLoading] = useState(true); // Gère le chargement
  const [error, setError] = useState(null); // Gère les erreurs

  useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/posts") // Requête vers l'API
      .then((response) => {
        if (!response.ok) {
          throw new Error("Erreur réseau"); // Vérifie si la réponse est correcte
        }
        return response.json(); // Convertit la réponse en JSON
      })
      .then((data) => {
        setData(data); // Met à jour les données
        setLoading(false); // Indique que le chargement est terminé
      })
      .catch((error) => {
        setError(error.message); // Gère les erreurs
        setLoading(false);
      });
  }, []); // Appel unique grâce à `[]`

  if (loading) return <p>Chargement...</p>; // Affiche un message de chargement
  if (error) return <p>Erreur : {error}</p>; // Affiche une erreur

  return (
    <div>
      <h1>Posts</h1>
      <ul>
        {data.map((post) => (
          <li key={post.id}>{post.title}</li> // Affiche la liste des posts
        ))}
      </ul>
    </div>
  );
};

export default App;
```

2. Utilisation d'`axios` :

`axios` est une bibliothèque externe qui simplifie les requêtes HTTP. Elle offre une gestion des erreurs et une syntaxe plus propre.

Exemple simplifié :

```
import React, { useEffect, useState } from "react";
import axios from "axios"; // Importation d'axios

const App = () => {
  const [data, setData] = useState([]); // Stocke les données
  const [loading, setLoading] = useState(true); // Gère le chargement
  const [error, setError] = useState(null); // Gère les erreurs

  useEffect(() => {
    axios
      .get("https://jsonplaceholder.typicode.com/posts") // Requête GET
      .then((response) => {
        setData(response.data); // Met à jour les données
        setLoading(false); // Indique que le chargement est terminé
      })
      .catch((error) => {
        setError(error.message); // Gère les erreurs
        setLoading(false);
      });
  }, []); // Appel unique grâce à `[]`

  if (loading) return <p>Chargement...</p>;
  if (error) return <p>Erreur : {error}</p>;

  return (
    <div>
      <h1>Posts</h1>
      <ul>
        {data.map((post) => (
          <li key={post.id}>{post.title}</li> // Affiche la liste des posts
        ))}
      </ul>
    </div>
  );
};

export default App;
```

Comparaison entre `fetch` et `axios` :

Critères	`fetch`	`axios`

Critères	`fetch`	`axios`
Disponibilité	Natif dans les navigateurs modernes	Nécessite une installation (`npm install axios`)
Gestion des erreurs	Manuelle (`response.ok`)	Automatique, rejette pour erreurs HTTP
Syntaxe	Plus longue pour les cas complexes	Plus concise et robuste
Support des navigateurs	Moderne (polyfill pour anciens navigateurs)	Compatible avec plus de navigateurs

Conclusion :

- `fetch` : Idéal pour des projets simples sans dépendances supplémentaires.
- `axios` : Préféré pour des projets complexes nécessitant une gestion avancée des requêtes.