useEffect, useState & generics i TypeScript

useEffect låter dig utföra sid-effekter, t.ex. datahämtning, efter att en komponent renderats.

Kodexempel:

```
useEffect(() => {
  // Denna funktion körs efter renderingen
  console.log("Komponenten har renderats");
}, []); // Tom dependencies-array: körs endast vid mount
```

Typescript och typer vid data-hämtning

Vi specificerar vilken typ av data vi förväntar oss genom en `type`:

```
type DataType = {
  id: number;
  title: string;
  //m.m.
};
```

UseState och generics

useState används ofta tillsammans med useEffect.

För att specificera den datatyp vi använder behöver vi använda generics med useState, t.ex:

```
//en array av vår ovan skapade datatyp ska användas
const [data, setData] = useState<DataType[]>([]);
```

Exempel: Datahämtning från API med useEffect, useState & Generics

I detta exempel:

- Vi använder useState för att lagra data och laddningstillstånd.
- Vi använder useEffect för att hämta data från ett API vid komponentens mount.
- Vi specificerar datatypen med en generisk typ (DataType[]).

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

type DataType = {
  id: number;
  title: string;
};
//komponenten:
const DataFetcher = () => {
  const [data, setData] = useState<DataType[]>([]);
  const [loading, setLoading] = useState<boolean>(true);

  useEffect(() => {
    //callback-funktion
    const fetchData = async () => {
      try {
        const response = await fetch("https://jsonplaceholder.typicode.com/posts");
        const result: DataType[] = await response.json();
        setData(result);
      } catch (error) {
        console.error("Fel vid hämtning av data:", error);
      }
    };

    fetchData();
  }, []); // Körs endast vid komponentens mount

  return (
    <div>
      {data.map((item) => (
        <div key={item.id}>
          <h3>{item.title}</h3>
        </div>
      ))}
    </div>
  );
};

export default DataFetcher;
```