"async/await" is a way to handle asynchronous operations in JavaScript. It's used to write asynchronous code that flows like synchronous code, making it easier to understand and maintain.

The **async** keyword is used to define an asynchronous function. An asynchronous function returns a Promise, which can be either resolved (fulfilled) or rejected. The **await** keyword is used inside an asynchronous function to pause its execution until a promise is resolved.

For example, if we want to get some data from an API, we can use **async/await** to wait for the response to arrive before moving on to the next line of code:

async function getData() {

const response = await fetch('https://api.example.com/data');

const data = await response.json();

return data;

}

In this example, the function **getData** is declared as **async**, and the **await** keyword is used to wait for the response from the API before moving on to parse the JSON data. This makes the code easier to read and understand, as it flows like synchronous code.

Object Destrucutring:

const Weathercard = ({tempinfo}) => { // as we are passing the array of the objects // again doing object destructuring. const [weatherState, setWeatherState] = useState(""); const { temp, humidity, pressure, weathermood, name, speed, country, sunset } =tempinfo; In the above code please explain the steps of the destructuring

In the code, destructuring is used to extract properties from the **tempinfo** object that is passed as a prop to the **Weathercard** component. The destructuring assignment syntax is used to extract the following properties from the **tempinfo** object:

1. **temp**
2. **humidity**
3. **pressure**
4. **weathermood**
5. **name**
6. **speed**
7. **country**
8. **sunset**

These properties are then stored in local variables with the same names as the properties, allowing the component to access their values without having to repeatedly reference the **tempinfo** object.

Additionally, the **useState** hook is used to define a state variable called **weatherState** with an initial value of an empty string. The hook returns an array with two elements, the current state value and a function to update the state, which are destructured and stored in **weatherState** and **setWeatherState** respectively.

Top of Form

Regenerate response

Bottom of Form