**Higher-Order Components (HOCs)**

Higher-Order Components (HOCs) in React are a pattern used for reusing component logic. A HOC is a function that takes a component and returns a new component with enhanced functionalities. This pattern is similar to how decorators work in other programming languages.

Here's an example to help illustrate the concept. We'll create a HOC that adds a loading spinner to a component while it's fetching data.

1. First, we define the HOC:

jsx

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

const withLoader = (WrappedComponent, url) => {

return (props) => {

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

const [loading, setLoading] = useState(true);

useEffect(() => {

const fetchData = async () => {

try {

const response = await fetch(url);

const result = await response.json();

setData(result);

} catch (error) {

console.error('Error fetching data:', error);

} finally {

setLoading(false);

}

};

fetchData();

}, [url]);

if (loading) {

return <div>Loading...</div>;

}

return <WrappedComponent data={data} {...props} />;

};

};

export default withLoader;

1. Next, we create a component that will be enhanced by the HOC:

jsx

import React from 'react';

const DataDisplay = ({ data }) => {

return (

<div>

<h1>Data Display</h1>

<pre>{JSON.stringify(data, null, 2)}</pre>

</div>

);

};

export default DataDisplay;

1. Finally, we wrap the component with the HOC and use it:

jsx

import React from 'react';

import withLoader from './withLoader';

import DataDisplay from './DataDisplay';

const DataDisplayWithLoader = withLoader(DataDisplay, 'https://api.example.com/data');

const App = () => {

return (

<div>

<DataDisplayWithLoader />

</div>

);

};

export default App;

In this example:

* We define the withLoader HOC, which fetches data from a given URL and manages the loading state.
* The HOC takes a WrappedComponent and a url as arguments and returns a new component that fetches data and passes it as a prop to the WrappedComponent.
* The DataDisplay component displays the fetched data.
* We wrap the DataDisplay component with the withLoader HOC to create a new component, DataDisplayWithLoader.
* The App component renders the DataDisplayWithLoader component, which shows a loading spinner while fetching data and then displays the data once it's fetched.