**Higher Order Components**

* A Higher-Order component (HOC) is an advanced technique in React for reusing component logic.
* Higher Order Component is a function that takes a component as input and returns a new component.
* Higher Order Components promote Code Reusability. We all know how important it is to develop reusable code.

Lets look at one Example. Assuming that we are developing an Admin Dashboard which will show Employee Data, Department Data and Project Data in the form of Reports.

We will have our REST API which will be giving us the data and the react application should display this data by fetching from the API.

Now lets Open Index.js file from our Demo Project using Visual Studio Code.

Here we will create three Component Classes. AdminDashboard, EmployeeReports and Department Reports. EmployeeReports Component will have the code to call the API, get the list of Employees and display them.

DepartmentReports Component will have the code to call the API, get the list of Departments  and display them.

AdminDashboard will be calling these two components and display the Reports.

And finally we will call the AdminDashboard Component and we will render it to our root container.

I have the Code handy and I am pasting it here. As this has been discussed already in our previous videos.

class EmployeeReports extends React.Component {

constructor(props) {

super(props);

this.state = {

employees: []

};

}

componentDidMount() {

fetch("https://localhost:44306/api/Employee")

.then(res => res.json())

.then(

(result) => {

this.setState({

employees: result

});

}

);

}

render() {

return (

<div>

<h2>Employees Data...</h2>

<table>

<thead>

<tr>

<th>Id</th>

<th>Name</th>

<th>Location</th>

<th>Salary</th>

</tr>

</thead>

<tbody>

{this.state.employees.map(emp => (

<tr key={emp.Id}>

<td>{emp.Id}</td>

<td>{emp.Name}</td>

<td>{emp.Location}</td>

<td>{emp.Salary}</td>

</tr>

))}

</tbody>

</table>

</div>

);

}

}

class DeptReports extends React.Component {

constructor(props) {

super(props);

this.state = {

dept: []

};

}

componentDidMount() {

fetch("https://localhost:44306/api/Dept")

.then(res => res.json())

.then(

(result) => {

this.setState({

dept: result

});

}

);

}

render() {

return (

<div>

<h2>Dept Data...</h2>

<table>

<thead>

<tr>

<th>Id</th>

<th>Name</th>

<th>Revenue</th>

</tr>

</thead>

<tbody>

{this.state.dept.map(d => (

<tr key={d.Id}>

<td>{d.Id}</td>

<td>{d.Name}</td>

<td>{d.Revenue}</td>

</tr>

))}

</tbody>

</table>

</div>

);

}

}

class AdminDashboard extends React.Component{

constructor(props){

super(props);

}

render(){

return (

<React.Fragment>

<EmployeeReports></EmployeeReports>

<DeptReports></DeptReports>

</React.Fragment>

);

}

}

const element=<AdminDashboard></AdminDashboard>

ReactDOM.render(element,document.getElementById("root"));

Now lets save these changes, navigate to the browser. We can see that both employee reports and department reports are displayed.

Now if we look at our EmployeeReports Component Class and DepartmentReports Component class, though they are sending a request to different Rest API, though they have different sets of columns and data that is getting rendered, on a high level they do the same job. Send API request, get the reports data and display the reports. We have to write the code on the similar lines even for showing Project Reports. Why cant we reuse the code instead of repeating the code again and again. This is where Higher Order Components comes into picture.

Remember that Higher-order component is a function that takes a component and returns a new component

Lets go ahead and create a function which accepts two inputs. First one is the Component and the second one is the input using which we will pass the information like the url and how many columns we want in the reports and any other component specific data.

As we know that Higher Order Component takes a Component as an input and returns a new Component, lets return a Component class

And with in class lets create a constructor and add one state object. To this state object add properties like url, columns and data. We initialize these properties by reading the data from the inputData parameter.

Lets implement ComponentDidMount method in which we will have the code to send the API request. I have the code and pasting it here.

We are sending the request and when we get the response we are updating our state object.

Now lets implement render method. With in this render method, we call another component to which we will pass this state data through a property and that component is responsible for displaying the data.

Now lets create this Data Component. Lets implement the render method.

Lets return a div container. Lets display the header data using h2 tag. We will access the header value through data property which is passed to this component.

Lets add a table, and the header columns are dependent on the columns which are passed through properties. Lets navigate through the columns array using map function and we will add the header.

Add table body and we have to display the data based on the number of records in the data which is passed to this component through properties. Lets loop through the data and for every row, we have to add the respective number of columns.

Lets create one Reports Component class and this component is going to be used as input component to our Higher Order Component.

Lets return div from the render method of this class.

Now we will create all the report component classes which are required. First lets start with EmployeeReports,

So we say const EmployeeReports=reportsHOC() and to this function, we will pass our Reports Component as input and we will pass another object which will contain the url, columns and header.

The Higher Order Component returns us a Component and we have assigned it to EmployeeReports. Now lets see how easy it becomes to generate DepartmentReports. We will call the reportsHOC function again and we will pass the other details through an Object.

Now we can repeat the same for any number of Report Components we want to generate.

Lets save these changes, navigate to the browser and we can see that both the Reports are being displayed.

Using HOC’s are common in any Enterprise application we develop using React.

import ReactDOM from "react-dom";

import React, { Component } from "react";

function reportsHOC(InputComponent, inputData) {

return class extends React.Component {

constructor(props) {

super(props);

this.state = {

data:[],

columns:inputData.columns,

header:inputData.header

};

}

componentDidMount() {

fetch(selectData.Url)

.then(res => res.json())

.then(

(result) => {

this.setState({

data: result

});

}

);

}

render() {

return (

<Data data={this.state}></Data>

);

}

};

}

class Data extends React.Component{

constructor(props){

super(props);

//alert(JSON.stringify(props));

}

render(){

return (

<div>

<h2>{this.props.data.header}...</h2>

<table>

<thead>

<tr>

{this.props.data.columns.map(c => (

<th>{c}</th>

))}

</tr>

</thead>

<tbody>

{this.props.data.data.map(emp => (

<tr key={emp.Id}>

{this.props.data.columns.map(c => (

<td>{emp[c]}</td>

))}

</tr>

))}

</tbody>

</table>

</div>

);

}

}

class Reports extends React.Component {

constructor(props) {

super(props);

}

render() {

return(

<div></div>

);

}

}

const EmployeeReports=reportsHOC(Employee,

{Url:'https://localhost:44306/api/Employee',

columns:['Id','Name','Location','Salary'],header:'Employee Data'});

const DeptReports=reportsHOC(Employee,{Url:'https://localhost:44306/api/Dept',

columns:['Id','Name','Revenue'],header:'Dept Data'});

class AdminDashboard extends React.Component{

constructor(props){

super(props);

}

render(){

return (

<React.Fragment>

<EmployeeReports></EmployeeReports>

<DeptReports></DeptReports>

</React.Fragment>

);

}

}

const element=<AdminDashboard></AdminDashboard>

ReactDOM.render(element,document.getElementById("root"));