Week-15 Solutions

-------------------------

Refactoring The ATM

----------------------------

index.html

--------------

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta

name="viewport"

content="width=device-width, initial-scale=1, shrink-to-fit=no"

/>

<meta name="theme-color" content="#000000" />

<!--

manifest.json provides metadata used when your web app is added to the

homescreen on Android. See https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/

-->

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

<link rel="shortcut icon" href="%PUBLIC\_URL%/favicon.ico" />

<link

rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"

/>

<script

src="https://unpkg.com/@babel/standalone/babel.min.js"

crossorigin

></script>

<title>Onclick Anything</title>

<!--

Notice the use of %PUBLIC\_URL% in the tags above.

It will be replaced with the URL of the `public` folder during the build.

Only files inside the `public` folder can be referenced from the HTML.

Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running `npm run build`.

-->

</head>

<body>

<noscript>

You need to enable JavaScript to run this app.

</noscript>

<div id="root"></div>

<!-- Load React. -->

<!-- Note: when deploying, replace "development.js" with "production.min.js". -->

<script

src="https://unpkg.com/react@17/umd/react.development.js"

crossorigin

></script>

<script

src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"

crossorigin

></script>

<!-- Load our React component. -->

<script type="text/jsx" src="./atm-improvements.jsx"></script>

</body>

</html>

atm-improvements.jsx

---------------------------

const ATMDeposit = ({ onChange, isDeposit, validTransaction }) => {

const choice = ["Deposit", "Cash Back"];

var flag;

var isValid = validTransaction;

if (isDeposit === true || isDeposit === false) {

flag = 1;

}

console.log(`ATM isDeposit: ${isDeposit}`);

return (

<div>

{flag && (

<label className="label huge">

<h3> {choice[Number(!isDeposit)]}</h3>

<input

id="number-input"

type="number"

width="200"

onChange={onChange}

></input>

<input

type="submit"

width="200"

value="Submit"

id="submit-input"

disabled={!isValid}

></input>

</label>

)}

</div>

);

};

const Account = () => {

const [deposit, setDeposit] = React.useState(0);

const [totalState, setTotalState] = React.useState(0);

const [isDeposit, setIsDeposit] = React.useState("");

const [atmMode, setAtmMode] = React.useState(true);

const [validTransaction, setValidTransaction] = React.useState(false);

let status = `Account Balance $ ${totalState} `;

console.log(`Account Rendered with isDeposit: ${isDeposit}`);

const handleChange = (event) => {

console.log(`handleChange ${event.target.value}`);

setDeposit(Number(event.target.value));

setValidTransaction(false);

if (event.target.value <= 0) {

return setValidTransaction(false);

}

if (atmMode === "Cash Back" && event.target.value > totalState) {

return setValidTransaction(false);

} else {

return setValidTransaction(true);

}

};

const handleSubmit = (event) => {

let newTotal = isDeposit ? totalState + deposit : totalState - deposit;

setTotalState(newTotal);

event.preventDefault();

};

const handleModeSelect = (event) => {

let selection = event.target.value;

setAtmMode(selection);

if (selection === "") setIsDeposit("");

if (selection === "Deposit") setIsDeposit(true);

if (selection === "Cash Back") setIsDeposit(false);

};

return (

<form onSubmit={handleSubmit}>

<h2 id="total">{status}</h2>

<label>Select an action below to continue</label>

<select

onChange={(e) => handleModeSelect(e)}

name="mode"

id="mode-select"

>

<option id="no-selection" value=""></option>

<option id="deposit-selection" value="Deposit">

Deposit

</option>

<option id="cashback-selection" value="Cash Back">

Cash Back

</option>

</select>

<ATMDeposit

onChange={handleChange}

isDeposit={isDeposit}

validTransaction={validTransaction}

></ATMDeposit>

</form>

);

};

// ========================================

ReactDOM.render(<Account />, document.getElementById("root"));

styles.css

-----------

#submit-input {

background-color:burlywood;

border: 6px ;

color: black;

border-radius: 25px;

width: 180px;

}

select {

display: inline-block;

margin-bottom: 20px;

background-color: lavender;

}

body {

background-color: lightpink;

}

#number-input {

width: 180px;

}

Output:

------------------

Fetch And Paginate Data

-----------------------------------

index.html

----------------

<html>

<head>

<meta charset="UTF-8" />

<title>React Cart</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"

integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous" />

<!-- Don't use this in production: -->

<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

</head>

<body>

<h1>React Get Data</h1>

<!-- We will put our React component inside this div. -->

<div id="root"></div>

<!-- Load React. -->

<script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>

<script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>

<script src="https://unpkg.com/react-router@5/umd/react-router.min.js" crossorigin></script>

<script src="https://unpkg.com/react-router-dom@5/umd/react-router-dom.min.js"></script>

<script src="https://unpkg.com/react-bootstrap@next/dist/react-bootstrap.min.js" crossorigin></script>

<!-- Load our Axios and our component. -->

<script src="https://unpkg.com/axios/dist/axios.min.js"></script>

<script src="FetchData.jsx" defer type="text/babel"></script>

</body>

</html>

FetchData.jsx

--------------------

const Pagination = ({ items, pageSize, onPageChange }) => {

const { Button } = ReactBootstrap;

if (items.length <= 1) return null;

let num = Math.ceil(items.length / pageSize);

let pages = range(1, num + 1);

const list = pages.map((page) => {

return (

<Button key={page} onClick={onPageChange} className="page-item">

{page}

</Button>

);

});

return (

<nav>

<ul className="pagination">{list}</ul>

</nav>

);

};

const range = (start, end) => {

return Array(end - start + 1)

.fill(0)

.map((item, i) => start + i);

};

function paginate(items, pageNumber, pageSize) {

const start = (pageNumber - 1) \* pageSize;

let page = items.slice(start, start + pageSize);

return page;

}

const useDataApi = (initialUrl, initialData) => {

const { useState, useEffect, useReducer } = React;

const [url, setUrl] = useState(initialUrl);

const [state, dispatch] = useReducer(dataFetchReducer, {

isLoading: false,

isError: false,

data: initialData,

});

useEffect(() => {

let didCancel = false;

const fetchData = async () => {

dispatch({ type: "FETCH\_INIT" });

try {

const result = await axios(url);

if (!didCancel) {

dispatch({ type: "FETCH\_SUCCESS", payload: result.data });

}

} catch (error) {

if (!didCancel) {

dispatch({ type: "FETCH\_FAILURE" });

}

}

};

fetchData();

return () => {

didCancel = true;

};

}, [url]);

return [state, setUrl];

};

const dataFetchReducer = (state, action) => {

switch (action.type) {

case "FETCH\_INIT":

return {

...state,

isLoading: true,

isError: false,

};

case "FETCH\_SUCCESS":

return {

...state,

isLoading: false,

isError: false,

data: action.payload,

};

case "FETCH\_FAILURE":

return {

...state,

isLoading: false,

isError: true,

};

default:

throw new Error();

}

};

// App that gets data from Hacker News url

function App() {

const { Fragment, useState, useEffect, useReducer } = React;

const [query, setQuery] = useState("MIT");

const [currentPage, setCurrentPage] = useState(1);

const pageSize = 10;

const [{ data, isLoading, isError }, doFetch] = useDataApi(

"https://hn.algolia.com/api/v1/search?query=MIT",

{

hits: [],

}

);

const handlePageChange = (e) => {

setCurrentPage(Number(e.target.textContent));

};

let page = data.hits;

if (page.length >= 1) {

page = paginate(page, currentPage, pageSize);

console.log(`currentPage: ${currentPage}`);

}

return (

<Fragment>

<form

onSubmit={(event) => {

doFetch(`http://hn.algolia.com/api/v1/search?query=${query}`);

event.preventDefault();

}}

>

<input

type="text"

value={query}

onChange={(event) => setQuery(event.target.value)}

/>

<button type="submit">Search</button>

</form>

{isError && <div>Something went wrong ...</div>}

{isLoading ? (

<div>Loading ...</div>

) : (

<ul>

{page.map((item) => (

<li key={item.objectID}>

<a href={item.url}>{item.title}</a>

</li>

))}

</ul>

)}

<Pagination

items={data.hits}

pageSize={pageSize}

onPageChange={handlePageChange}

></Pagination>

</Fragment>

);

}

// ========================================

ReactDOM.render(<App />, document.getElementById("root"));

Output

---------------

Knowledge Check 15.1: Fetching And Rendering Data

----------------------------------------------------------------------------

1. (3) (4) (2) (1)

2. All UseState -prefixed setFunction() will trigger a re-rendering of the website

3. (a) (d)

4. (c) (d)