Looping elements in react with passing props

App.js

import ExpenseItem from "./components/ExpenseItem";

function App() {

  const expenses = [

    {

      id: "e1",

      title: "Toilet Paper",

      amount: 94.12,

      date: new Date(2020, 7, 14),

    },

    { id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },

    {

      id: "e3",

      title: "Car Insurance",

      amount: 294.67,

      date: new Date(2021, 2, 28),

    },

    {

      id: "e4",

      title: "New Desk (Wooden)",

      amount: 450,

      date: new Date(2021, 5, 12),

    },

  ];

  const expenseItems = expenses.map((exp) => (

    <ExpenseItem

      key={exp.id}

      title={exp.title}

      amount={exp.amount}

      date={exp.date}

    ></ExpenseItem>

  ));

  return (

    <div>

      <h2>Let's get started</h2>

      {expenseItems}

    </div>

  );

}

export default App;

ExpenseItem.js

import "./ExpenseItem.css";

function ExpenseItem(props) {

  return (

    <div className="expense-item">

      <div>{props.date.toISOString()}</div>

      <div className="expense-item\_\_description">

        <h2>{props.title}</h2>

        <div className="expense-item\_\_price">${props.amount}</div>

      </div>

    </div>

  );

}

export default ExpenseItem;

useState

import React, { useState } from 'react';

import './ExpenseForm.css';

const ExpenseForm = () => {

const [enteredTitle, setEnteredTitle] = useState('');

const [enteredAmount, setEnteredAmount] = useState('');

const [enteredDate, setEnteredDate] = useState('');

// const [userInput, setUserInput] = useState({

// enteredTitle: '',

// enteredAmount: '',

// enteredDate: '',

// });

const titleChangeHandler = (event) => {

setEnteredTitle(event.target.value);

// setUserInput({

// ...userInput,

// enteredTitle: event.target.value,

// });

//it make sure always get the latest snapshot of

// of previous state

// setUserInput((prevState) => {

// return { ...prevState, enteredTitle: event.target.value };

// });

};

const amountChangeHandler = (event) => {

setEnteredAmount(event.target.value);

// setUserInput({

// ...userInput,

// enteredAmount: event.target.value,

// });

};

const dateChangeHandler = (event) => {

setEnteredDate(event.target.value);

// setUserInput({

// ...userInput,

// enteredDate: event.target.value,

// });

};

return (

<form>

<div className='new-expense\_\_controls'>

<div className='new-expense\_\_control'>

<label>Title</label>

<input type='text' onChange={titleChangeHandler} />

</div>

<div className='new-expense\_\_control'>

<label>Amount</label>

<input

type='number'

min='0.01'

step='0.01'

onChange={amountChangeHandler}

/>

</div>

<div className='new-expense\_\_control'>

<label>Date</label>

<input

type='date'

min='2019-01-01'

max='2022-12-31'

onChange={dateChangeHandler}

/>

</div>

</div>

<div className='new-expense\_\_actions'>

<button type='submit'>Add Expense</button>

</div>

</form>

);

};

export default ExpenseForm;

Dynamically set style

<div

className='chart-bar\_\_fill'

style={{ 'background-color': 'red', height: barFillHeight }}

></div>

or

<div

className='chart-bar\_\_fill'

style={{ backgroundColor: 'red', height: barFillHeight }}

></div>

Style component

it is global applied css

import './Button.css';

Inline styling

import React, { useState } from 'react';

import Button from '../../UI/Button/Button';

import './CourseInput.css';

const CourseInput = (props) => {

const [enteredValue, setEnteredValue] = useState('');

const [isValid, setIsvalid] = useState(true);

const goalInputChangeHandler = (event) => {

if (event.target.value.trim().length > 0) {

setIsvalid(true);

}

setEnteredValue(event.target.value);

};

const formSubmitHandler = (event) => {

event.preventDefault();

if (enteredValue.trim().length === 0) {

setIsvalid(false);

return;

}

props.onAddGoal(enteredValue);

setEnteredValue('');

};

return (

<form onSubmit={formSubmitHandler}>

<div className='form-control'>

<label style={{ color: !isValid ? 'red' : 'black' }}>Course Goal</label>

<input

type='text'

value={enteredValue}

style={{

backgroundColor: !isValid ? 'salmon' : 'transparent',

borderColor: !isValid ? 'red' : '#ccc',

}}

onChange={goalInputChangeHandler}

/>

</div>

<Button type='submit'>Add Goal</Button>

</form>

);

};

export default CourseInput;

**setting css classes dynamically**

import React, { useState } from 'react';

import Button from '../../UI/Button/Button';

import './CourseInput.css';

const CourseInput = (props) => {

const [enteredValue, setEnteredValue] = useState('');

const [isValid, setIsvalid] = useState(true);

const goalInputChangeHandler = (event) => {

if (event.target.value.trim().length > 0) {

setIsvalid(true);

}

setEnteredValue(event.target.value);

};

const formSubmitHandler = (event) => {

event.preventDefault();

if (enteredValue.trim().length === 0) {

setIsvalid(false);

return;

}

props.onAddGoal(enteredValue);

setEnteredValue('');

};

return (

<form onSubmit={formSubmitHandler}>

<div className={`form-control ${!isValid ? 'invalid' : ''}`}>

<label>Course Goal</label>

<input

type='text'

value={enteredValue}

onChange={goalInputChangeHandler}

/>

</div>

<Button type='submit'>Add Goal</Button>

</form>

);

};

export default CourseInput;

**Styled components**

import React from 'react';

import styled from 'styled-components';

//it automatically apply other props such as props.onClick and props.type

const Button = styled.button`

font: inherit;

padding: 0.5rem 1.5rem;

border: 1px solid #8b005d;

color: white;

background: #8b005d;

box-shadow: 0 0 4px rgba(0, 0, 0, 0.26);

cursor: pointer;

&:focus {

outline: none;

}

&:hover,

&:active {

background: #ac0e77;

border-color: #ac0e77;

box-shadow: 0 0 8px rgba(0, 0, 0, 0.26);

}

`;

// const Button = (props) => {

// return (

// <button type={props.type} className='button' onClick={props.onClick}>

// {props.children}

// </button>

// );

// };

export default Button;

Styled component with dynamic props

import React, { useState } from 'react';

import styled from 'styled-components';

import Button from '../../UI/Button/Button';

import './CourseInput.css';

const FormControl = styled.div`

margin: 0.5rem 0;

& label {

font-weight: bold;

display: block;

margin-bottom: 0.5rem;

color: ${(props) => (props.invalid ? 'red' : 'black')};

}

& input {

display: block;

width: 100%;

border: 1px solid ${(props) => (props.invalid ? 'red' : '#ccc')};

background: ${(props) => (props.invalid ? '#ffd7d7' : 'transparent')};

font: inherit;

line-height: 1.5rem;

padding: 0 0.25rem;

}

& input:focus {

outline: none;

background: #fad0ec;

border-color: #8b005d;

}

`;

const CourseInput = (props) => {

const [enteredValue, setEnteredValue] = useState('');

const [isValid, setIsValid] = useState(true);

const goalInputChangeHandler = (event) => {

if (event.target.value.trim().length > 0) {

setIsValid(true);

}

setEnteredValue(event.target.value);

};

const formSubmitHandler = (event) => {

event.preventDefault();

if (enteredValue.trim().length === 0) {

setIsValid(false);

return;

}

props.onAddGoal(enteredValue);

};

return (

<form onSubmit={formSubmitHandler}>

<FormControl invalid={!isValid}>

<label>Course Goal</label>

<input type='text' onChange={goalInputChangeHandler} />

</FormControl>

<Button type='submit'>Add Goal</Button>

</form>

);

};

export default CourseInput;

or pass classname dynamically

import React, { useState } from 'react';

import styled from 'styled-components';

import Button from '../../UI/Button/Button';

import './CourseInput.css';

const FormControl = styled.div`

margin: 0.5rem 0;

& label {

font-weight: bold;

display: block;

margin-bottom: 0.5rem;

}

& input {

display: block;

width: 100%;

border: 1px solid #ccc;

font: inherit;

line-height: 1.5rem;

padding: 0 0.25rem;

}

& input:focus {

outline: none;

background: #fad0ec;

border-color: #8b005d;

}

&.invalid label {

color: red;

}

&.invalid input {

border-color: red;

background: rgb(235, 79, 79);

}

`;

const CourseInput = (props) => {

const [enteredValue, setEnteredValue] = useState('');

const [isValid, setIsValid] = useState(true);

const goalInputChangeHandler = (event) => {

if (event.target.value.trim().length > 0) {

setIsValid(true);

}

setEnteredValue(event.target.value);

};

const formSubmitHandler = (event) => {

event.preventDefault();

if (enteredValue.trim().length === 0) {

setIsValid(false);

return;

}

props.onAddGoal(enteredValue);

};

return (

<form onSubmit={formSubmitHandler}>

<FormControl className={!isValid && 'invalid'}>

<label>Course Goal</label>

<input type='text' onChange={goalInputChangeHandler} />

</FormControl>

<Button type='submit'>Add Goal</Button>

</form>

);

};

export default CourseInput;

style component with media queries

import React from 'react';

import styled from 'styled-components';

//it automatically apply other props such as props.onClick and props.type

const Button = styled.button`

width: 100%;

font: inherit;

padding: 0.5rem 1.5rem;

border: 1px solid #8b005d;

color: white;

background: #8b005d;

box-shadow: 0 0 4px rgba(0, 0, 0, 0.26);

cursor: pointer;

@media (min-width: 768px) {

width: auto;

}

&:focus {

outline: none;

}

&:hover,

&:active {

background: #ac0e77;

border-color: #ac0e77;

box-shadow: 0 0 8px rgba(0, 0, 0, 0.26);

}

`;

// const Button = (props) => {

// return (

// <button type={props.type} className='button' onClick={props.onClick}>

// {props.children}

// </button>

// );

// };

export default Button;

**// Note**  styled component automatically pass props and event handler and props.children

Css Modules

we have to rename the css file eg persons.module.css

import React, { useState } from 'react';

import Button from '../../UI/Button/Button';

import styles from './CourseInput.module.css';

const CourseInput = props => {

const [enteredValue, setEnteredValue] = useState('');

const [isValid, setIsValid] = useState(true);

const goalInputChangeHandler = event => {

if (event.target.value.trim().length > 0) {

setIsValid(true);

}

setEnteredValue(event.target.value);

};

const formSubmitHandler = event => {

event.preventDefault();

if (enteredValue.trim().length === 0) {

setIsValid(false);

return;

}

props.onAddGoal(enteredValue);

};

return (

<form onSubmit={formSubmitHandler}>

//note how to apply classes

<div

className={`${styles['form-control']} ${!isValid && styles.invalid}`}

>

<label>Course Goal</label>

<input type="text" onChange={goalInputChangeHandler} />

</div>

<Button type="submit">Add Goal</Button>

</form>

);

};

export default CourseInput;

Debugging

**Handling Extra div in JSX**

solution 1

Wrap with extra div

solution wrap all the code inside the array and give key to each component

import React, { useState } from 'react';

import Card from '../UI/Card';

import Button from '../UI/Button';

import ErrorModal from '../UI/ErrorModal';

import classes from './AddUser.module.css';

const AddUser = (props) => {

const [userData, setUserData] = useState({

enteredUsername: '',

enteredAge: '',

});

const [error, setError] = useState();

const addUserHandler = (event) => {

event.preventDefault();

if (

userData.enteredUsername.trim().length === 0 ||

userData.enteredAge.trim().length === 0

) {

setError({

title: 'Invalid input',

message: 'Please enter a valid name and age (non-empty values).',

});

return;

}

if (+userData.enteredAge < 1) {

setError({

title: 'Invalid age',

message: 'Please enter a valid age (> 0).',

});

return;

}

props.onAddUser(userData.enteredUsername, userData.enteredAge);

setUserData({

enteredUsername: '',

enteredAge: '',

});

};

const usernameChangeHandler = (event) => {

setUserData((prevdata) => {

return { ...prevdata, enteredUsername: event.target.value };

});

};

const ageChangeHandler = (event) => {

setUserData((prevdata) => {

return { ...prevdata, enteredAge: event.target.value };

});

};

const errorHandler = () => {

setError(null);

};

//wrap component inside the array

return [

error && (

<ErrorModal

key='errorHandler'

title={error.title}

message={error.message}

onConfirm={errorHandler}

/>

),

<Card className={classes.input} key='form-card-input'>

<form onSubmit={addUserHandler}>

<label htmlFor='username'>Username</label>

<input

id='username'

type='text'

value={userData.enteredUsername}

onChange={usernameChangeHandler}

/>

<label htmlFor='age'>Age (Years)</label>

<input

id='age'

type='number'

value={userData.enteredAge}

onChange={ageChangeHandler}

/>

<Button type='submit'>Add User</Button>

</form>

</Card>,

];

};

export default AddUser;

Solution 3

Create wrapper component

export const Wrapper = (props) => {

return props.children;

};

Wrap the entire code with Wrapper component

import React, { useState } from 'react';

import Card from '../UI/Card';

import Button from '../UI/Button';

import ErrorModal from '../UI/ErrorModal';

import classes from './AddUser.module.css';

import { Wrapper } from '../UI/Helper/Wrapper';

const AddUser = (props) => {

return (

<Wrapper>

{error && (

<ErrorModal

title={error.title}

message={error.message}

onConfirm={errorHandler}

/>

)}

<Card className={classes.input}>

<form onSubmit={addUserHandler}>

<label htmlFor='username'>Username</label>

<input

id='username'

type='text'

value={userData.enteredUsername}

onChange={usernameChangeHandler}

/>

<label htmlFor='age'>Age (Years)</label>

<input

id='age'

type='number'

value={userData.enteredAge}

onChange={ageChangeHandler}

/>

<Button type='submit'>Add User</Button>

</form>

</Card>

</Wrapper>

);

};

export default AddUser;

Using Fragment

import React, { useState } from 'react';

function App() {

return (

<>

<AddUser onAddUser={addUserHandler} />

<UsersList users={usersList} />

</>

);

}

export default App;

or

import React, { useState } from 'react';

return (

<React.Fragment>

<AddUser onAddUser={addUserHandler} />

<UsersList users={usersList} />

</React.Fragment>

);

or

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

return (

<Fragment>

<AddUser onAddUser={addUserHandler} />

<UsersList users={usersList} />

</Fragment>

);

React Portal

It is generally used to replace the position of the render component inside the dom. It makes sure the correct position of the dom element makes our semantically html clean code. Eg: generally we use Modal and Sidebar at the top level or near to the body tag.

Public/index.html

define div where we want to place our component

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

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

<meta name="viewport" content="width=device-width, initial-scale=1" />

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

<meta

name="description"

content="Web site created using create-react-app"

/>

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

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

<title>React App</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<!- div define here-->

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

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

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

</body>

</html>

ErrorModal.js

import React from 'react';

import ReactDom from 'react-dom'; //import ReactDom

import Card from './Card';

import Button from './Button';

import classes from './ErrorModal.module.css';

const Backdrop = (props) => {

return <div className={classes.backdrop} onClick={props.onConfirm} />;

};

const ModalOverlay = (props) => {

return (

<Card className={classes.modal}>

<header className={classes.header}>

<h2>{props.title}</h2>

</header>

<div className={classes.content}>

<p>{props.message}</p>

</div>

<footer className={classes.actions}>

<Button onClick={props.onConfirm}>Okay</Button>

</footer>

</Card>

);

};

const ErrorModal = (props) => {

return (

<React.Fragment>

//pass this component inside the dom

{ReactDom.createPortal(

<Backdrop onConfirm={props.onConfirm} />,

document.getElementById('backdrop-root')

)}

{ReactDom.createPortal(

<ModalOverlay

title={props.title}

message={props.message}

onConfirm={props.onConfirm}

/>,

document.getElementById('overlay-root')

)}

</React.Fragment>

);

};

export default ErrorModal;

Use Ref

It is used to access the dom element. Use in the rear scenario . Try to manage your state with react. While we are using ref its state is not managed by react. It is recommended dont manipulate the DOM by ref. But reading data is a good use of useref.

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

import Card from '../UI/Card';

import Button from '../UI/Button';

import ErrorModal from '../UI/ErrorModal';

import classes from './AddUser.module.css';

const AddUser = (props) => {

const inputAgeRef = useRef();

const inputNameRef = useRef();

const [error, setError] = useState();

const addUserHandler = (event) => {

event.preventDefault();

if (

inputNameRef.current.value.trim().length === 0 ||

inputAgeRef.current.value.trim().length === 0

) {

setError({

title: 'Invalid input',

message: 'Please enter a valid name and age (non-empty values).',

});

return;

}

if (+inputAgeRef.current.value < 1) {

setError({

title: 'Invalid age',

message: 'Please enter a valid age (> 0).',

});

return;

}

props.onAddUser(inputNameRef.current.value, inputAgeRef.current.value);

inputNameRef.current.value = '';

inputAgeRef.current.value = '';

};

const errorHandler = () => {

setError(null);

};

return (

<React.Fragment>

{error && (

<ErrorModal

title={error.title}

message={error.message}

onConfirm={errorHandler}

/>

)}

<Card className={classes.input}>

<form onSubmit={addUserHandler}>

<label htmlFor='username'>Username</label>

<input id='username' type='text' ref={inputNameRef} />

<label htmlFor='age'>Age (Years)</label>

<input id='age' type='number' ref={inputAgeRef} />

<Button type='submit'>Add User</Button>

</form>

</Card>

</React.Fragment>

);

};

export default AddUser;

Controlled and Uncontrolled Component

State manage by react is called controlled component eg useState

State is not manage by react is called uncontrolled component eg useRef

**Handling Side Effect**

it run the code when component is mounted

const [isLoggedIn, setIsLoggedIn] = useState(false);

useEffect(() => {

const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');

if (storedUserLoggedInInformation === '1') {

setIsLoggedIn(true);

}

console.log('Render App.js');

}, []);

without [] it will run the code every time

useEffect with dependency

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

import Card from '../UI/Card/Card';

import classes from './Login.module.css';

import Button from '../UI/Button/Button';

const Login = (props) => {

const [enteredEmail, setEnteredEmail] = useState('');

const [emailIsValid, setEmailIsValid] = useState();

const [enteredPassword, setEnteredPassword] = useState('');

const [passwordIsValid, setPasswordIsValid] = useState();

const [formIsValid, setFormIsValid] = useState(false);

//it call when first time render and enteredEmail,enteredPasssword is //changed

useEffect(() => {

setFormIsValid(

enteredEmail.includes('@') && enteredPassword.trim().length > 6

);

}, [enteredEmail, enteredPassword]);

const emailChangeHandler = (event) => {

setEnteredEmail(event.target.value);

};

const passwordChangeHandler = (event) => {

setEnteredPassword(event.target.value);

};

const validateEmailHandler = () => {

setEmailIsValid(enteredEmail.includes('@'));

};

const validatePasswordHandler = () => {

setPasswordIsValid(enteredPassword.trim().length > 6);

};

const submitHandler = (event) => {

event.preventDefault();

props.onLogin(enteredEmail, enteredPassword);

};

return (

<Card className={classes.login}>

<form onSubmit={submitHandler}>

<div

className={`${classes.control} ${

emailIsValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='email'>E-Mail</label>

<input

type='email'

id='email'

value={enteredEmail}

onChange={emailChangeHandler}

onBlur={validateEmailHandler}

/>

</div>

<div

className={`${classes.control} ${

passwordIsValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='password'>Password</label>

<input

type='password'

id='password'

value={enteredPassword}

onChange={passwordChangeHandler}

onBlur={validatePasswordHandler}

/>

</div>

<div className={classes.actions}>

<Button type='submit' className={classes.btn} disabled={!formIsValid}>

Login

</Button>

</div>

</form>

</Card>

);

};

export default Login;

don’t use as dependency

setterState function

window and other global object

variable and function define outside the component

Only add as dependency

variableState that is change inside the useEffect

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

import Card from '../UI/Card/Card';

import classes from './Login.module.css';

import Button from '../UI/Button/Button';

const Login = (props) => {

const [enteredEmail, setEnteredEmail] = useState('');

const [emailIsValid, setEmailIsValid] = useState();

const [enteredPassword, setEnteredPassword] = useState('');

const [passwordIsValid, setPasswordIsValid] = useState();

const [formIsValid, setFormIsValid] = useState(false);

useEffect(() => {

//1)execute first

const identifier = setTimeout(() => {

console.log('Login.js check form is valid');

setFormIsValid(

enteredEmail.includes('@') && enteredPassword.trim().length > 6

);

}, 500);

//2)Excute first in second render

//3)Execute when component unmounted

return () => {

console.log('Cleanup');

clearTimeout(identifier);

};

}, [enteredEmail, enteredPassword]);

const emailChangeHandler = (event) => {

setEnteredEmail(event.target.value);

};

const passwordChangeHandler = (event) => {

setEnteredPassword(event.target.value);

};

const validateEmailHandler = () => {

setEmailIsValid(enteredEmail.includes('@'));

};

const validatePasswordHandler = () => {

setPasswordIsValid(enteredPassword.trim().length > 6);

};

const submitHandler = (event) => {

event.preventDefault();

props.onLogin(enteredEmail, enteredPassword);

};

return (

<Card className={classes.login}>

<form onSubmit={submitHandler}>

<div

className={`${classes.control} ${

emailIsValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='email'>E-Mail</label>

<input

type='email'

id='email'

value={enteredEmail}

onChange={emailChangeHandler}

onBlur={validateEmailHandler}

/>

</div>

<div

className={`${classes.control} ${

passwordIsValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='password'>Password</label>

<input

type='password'

id='password'

value={enteredPassword}

onChange={passwordChangeHandler}

onBlur={validatePasswordHandler}

/>

</div>

<div className={classes.actions}>

<Button type='submit' className={classes.btn} disabled={!formIsValid}>

Login

</Button>

</div>

</form>

</Card>

);

};

export default Login;

useEffect another Example

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

import Card from '../UI/Card/Card';

import classes from './Login.module.css';

import Button from '../UI/Button/Button';

const Login = (props) => {

const [enteredEmail, setEnteredEmail] = useState('');

const [emailIsValid, setEmailIsValid] = useState();

const [enteredPassword, setEnteredPassword] = useState('');

const [passwordIsValid, setPasswordIsValid] = useState();

const [formIsValid, setFormIsValid] = useState(false);

useEffect(() => {

console.log('Run in every render cycle');

});

useEffect(() => {

console.log('Run on first render cycle only ');

}, []);

//Run on First render and on dependecy change

useEffect(() => {

//1)execute first

const identifier = setTimeout(() => {

console.log('Login.js check form is valid');

setFormIsValid(

enteredEmail.includes('@') && enteredPassword.trim().length > 6

);

}, 500);

//2)Excute first in second render

//3)Execute when component unmounted

return () => {

console.log('Cleanup');

clearTimeout(identifier);

};

}, [enteredEmail, enteredPassword]);

const emailChangeHandler = (event) => {

setEnteredEmail(event.target.value);

};

const passwordChangeHandler = (event) => {

setEnteredPassword(event.target.value);

};

const validateEmailHandler = () => {

setEmailIsValid(enteredEmail.includes('@'));

};

const validatePasswordHandler = () => {

setPasswordIsValid(enteredPassword.trim().length > 6);

};

const submitHandler = (event) => {

event.preventDefault();

props.onLogin(enteredEmail, enteredPassword);

};

return (

<Card className={classes.login}>

<form onSubmit={submitHandler}>

<div

className={`${classes.control} ${

emailIsValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='email'>E-Mail</label>

<input

type='email'

id='email'

value={enteredEmail}

onChange={emailChangeHandler}

onBlur={validateEmailHandler}

/>

</div>

<div

className={`${classes.control} ${

passwordIsValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='password'>Password</label>

<input

type='password'

id='password'

value={enteredPassword}

onChange={passwordChangeHandler}

onBlur={validatePasswordHandler}

/>

</div>

<div className={classes.actions}>

<Button type='submit' className={classes.btn} disabled={!formIsValid}>

Login

</Button>

</div>

</form>

</Card>

);

};

export default Login;

**useReducer**

const [state, dispatch] = useReducer(reducer, initialArg, init);

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

import Card from '../UI/Card/Card';

import classes from './Login.module.css';

import Button from '../UI/Button/Button';

const emailReducer = (prevState, action) => {

if (action.type === 'USER\_INPUT') {

return { value: action.val, isValid: action.val.includes('@') };

}

if (action.type === 'INPUT\_BLUR') {

return { value: prevState.value, isValid: prevState.value.includes('@') };

}

return { value: '', isValid: false };

};

const passwordReducer = (preState, action) => {

if (action.type === 'USER\_INPUT') {

return {

value: action.val,

isValid: action.val.trim().length > 6,

};

}

if (action.type === 'INPUT\_BLUR') {

return {

value: preState.value,

isValid: preState.value.trim().length > 6,

};

}

return { value: '', isValid: false };

};

const Login = (props) => {

// const [enteredEmail, setEnteredEmail] = useState('');

// const [emailIsValid, setEmailIsValid] = useState();

// const [enteredPassword, setEnteredPassword] = useState('');

// const [passwordIsValid, setPasswordIsValid] = useState();

const [formIsValid, setFormIsValid] = useState(false);

const [emailState, dispatchEmail] = useReducer(emailReducer, {

value: '',

isValid: null,

});

const [passwordState, dispatchPassword] = useReducer(passwordReducer, {

value: '',

isValid: null,

});

useEffect(() => {

const identifier = setTimeout(() => {

console.log('Login.js check form is valid');

setFormIsValid(emailState.isValid && passwordState.isValid);

}, 500);

return () => {

console.log('Cleanup');

clearTimeout(identifier);

};

}, [emailState.isValid, passwordState.isValid]);

const emailChangeHandler = (event) => {

dispatchEmail({

type: 'USER\_INPUT',

val: event.target.value,

});

};

const passwordChangeHandler = (event) => {

//setEnteredPassword(event.target.value);

dispatchPassword({

type: 'USER\_INPUT',

val: event.target.value,

});

};

const validateEmailHandler = () => {

// setEmailIsValid(enteredEmail.includes('@'));

dispatchEmail({

type: 'INPUT\_BLUR',

});

};

const validatePasswordHandler = () => {

//setPasswordIsValid(enteredPassword.trim().length > 6);

dispatchPassword({

type: 'INPUT\_BLUR',

});

};

const submitHandler = (event) => {

event.preventDefault();

props.onLogin(emailState.value, passwordState.value);

};

return (

<Card className={classes.login}>

<form onSubmit={submitHandler}>

<div

className={`${classes.control} ${

emailState.isValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='email'>E-Mail</label>

<input

type='email'

id='email'

value={emailState.value}

onChange={emailChangeHandler}

onBlur={validateEmailHandler}

/>

</div>

<div

className={`${classes.control} ${

passwordState.isValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='password'>Password</label>

<input

type='password'

id='password'

value={passwordState.value}

onChange={passwordChangeHandler}

onBlur={validatePasswordHandler}

/>

</div>

<div className={classes.actions}>

<Button type='submit' className={classes.btn} disabled={!formIsValid}>

Login

</Button>

</div>

</form>

</Card>

);

};

export default Login;

Context in react

auth-context.js

import React from 'react';

//react.createContext return component;

const AuthContext = React.createContext({

isLoggedIn: false,

loggedOut: () => {},

loggedIn: () => {},

});

export default AuthContext;

app.js

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

import Login from './components/Login/Login';

import Home from './components/Home/Home';

import MainHeader from './components/MainHeader/MainHeader';

import AuthContext from './store/auth-context';

function App() {

const [isLoggedIn, setIsLoggedIn] = useState(false);

useEffect(() => {

const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');

if (storedUserLoggedInInformation === '1') {

setIsLoggedIn(true);

}

console.log('Render App.js');

}, []);

const loginHandler = (email, password) => {

localStorage.setItem('isLoggedIn', '1');

setIsLoggedIn(true);

};

const logoutHandler = () => {

localStorage.removeItem('isLoggedIn');

setIsLoggedIn(false);

};

return (

<React.Fragment>

<AuthContext.Provider

value={{

isLoggedIn: isLoggedIn,

loggedOut: logoutHandler,

loggedIn: loginHandler,

}}

>

<MainHeader />

<main>

{!isLoggedIn && <Login />}

{isLoggedIn && <Home />}

</main>

</AuthContext.Provider>

</React.Fragment>

);

}

export default App;

navigation.js

import React from 'react';

import AuthContext from '../../store/auth-context';

import classes from './Navigation.module.css';

const Navigation = (props) => {

return (

<AuthContext.Consumer>

{(ctx) => {

return (

<nav className={classes.nav}>

<ul>

{ctx.isLoggedIn && (

<li>

<a href='/'>Users</a>

</li>

)}

{ctx.isLoggedIn && (

<li>

<a href='/'>Admin</a>

</li>

)}

{ctx.isLoggedIn && (

<li>

<button onClick={ctx.loggedOut}>Logout</button>

</li>

)}

</ul>

</nav>

);

}}

</AuthContext.Consumer>

);

};

export default Navigation;

login.js

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

import Card from '../UI/Card/Card';

import classes from './Login.module.css';

import Button from '../UI/Button/Button';

import AuthContext from '../../store/auth-context';

const emailReducer = (prevState, action) => {

if (action.type === 'USER\_INPUT') {

return { value: action.val, isValid: action.val.includes('@') };

}

if (action.type === 'INPUT\_BLUR') {

return { value: prevState.value, isValid: prevState.value.includes('@') };

}

return { value: '', isValid: false };

};

const passwordReducer = (preState, action) => {

if (action.type === 'USER\_INPUT') {

return {

value: action.val,

isValid: action.val.trim().length > 6,

};

}

if (action.type === 'INPUT\_BLUR') {

return {

value: preState.value,

isValid: preState.value.trim().length > 6,

};

}

return { value: '', isValid: false };

};

const Login = (props) => {

// const [enteredEmail, setEnteredEmail] = useState('');

// const [emailIsValid, setEmailIsValid] = useState();

// const [enteredPassword, setEnteredPassword] = useState('');

// const [passwordIsValid, setPasswordIsValid] = useState();

const [formIsValid, setFormIsValid] = useState(false);

const [emailState, dispatchEmail] = useReducer(emailReducer, {

value: '',

isValid: null,

});

const [passwordState, dispatchPassword] = useReducer(passwordReducer, {

value: '',

isValid: null,

});

useEffect(() => {

const identifier = setTimeout(() => {

console.log('Login.js check form is valid');

setFormIsValid(emailState.isValid && passwordState.isValid);

}, 500);

return () => {

console.log('Cleanup');

clearTimeout(identifier);

};

}, [emailState.isValid, passwordState.isValid]);

const emailChangeHandler = (event) => {

dispatchEmail({

type: 'USER\_INPUT',

val: event.target.value,

});

};

const passwordChangeHandler = (event) => {

//setEnteredPassword(event.target.value);

dispatchPassword({

type: 'USER\_INPUT',

val: event.target.value,

});

};

const validateEmailHandler = () => {

// setEmailIsValid(enteredEmail.includes('@'));

dispatchEmail({

type: 'INPUT\_BLUR',

});

};

const validatePasswordHandler = () => {

//setPasswordIsValid(enteredPassword.trim().length > 6);

dispatchPassword({

type: 'INPUT\_BLUR',

});

};

const submitHandler = (event, loggedInFn) => {

event.preventDefault();

loggedInFn(emailState.value, passwordState.value);

// props.onLogin(emailState.value, passwordState.value);

};

return (

<AuthContext.Consumer>

{(ctx) => {

return (

<Card className={classes.login}>

<form onSubmit={(event) => submitHandler(event, ctx.loggedIn)}>

<div

className={`${classes.control} ${

emailState.isValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='email'>E-Mail</label>

<input

type='email'

id='email'

value={emailState.value}

onChange={emailChangeHandler}

onBlur={validateEmailHandler}

/>

</div>

<div

className={`${classes.control} ${

passwordState.isValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='password'>Password</label>

<input

type='password'

id='password'

value={passwordState.value}

onChange={passwordChangeHandler}

onBlur={validatePasswordHandler}

/>

</div>

<div className={classes.actions}>

<Button

type='submit'

className={classes.btn}

disabled={!formIsValid}

>

Login

</Button>

</div>

</form>

</Card>

);

}}

</AuthContext.Consumer>

);

};

export default Login;

useContext hook

it is use to pass the data among the components

app.js

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

import Login from './components/Login/Login';

import Home from './components/Home/Home';

import MainHeader from './components/MainHeader/MainHeader';

import AuthContext from './store/auth-context';

function App() {

const [isLoggedIn, setIsLoggedIn] = useState(false);

useEffect(() => {

const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');

if (storedUserLoggedInInformation === '1') {

setIsLoggedIn(true);

}

console.log('Render App.js');

}, []);

const loginHandler = (email, password) => {

localStorage.setItem('isLoggedIn', '1');

setIsLoggedIn(true);

};

const logoutHandler = () => {

localStorage.removeItem('isLoggedIn');

setIsLoggedIn(false);

};

return (

<React.Fragment>

<AuthContext.Provider

value={{

isLoggedIn: isLoggedIn,

loggedOut: logoutHandler,

loggedIn: loginHandler,

}}

>

<MainHeader />

<main>

{!isLoggedIn && <Login />}

{isLoggedIn && <Home />}

</main>

</AuthContext.Provider>

</React.Fragment>

);

}

export default App;

navigation.js

import React, { useContext } from 'react';

import AuthContext from '../../store/auth-context';

import classes from './Navigation.module.css';

const Navigation = (props) => {

const ctx = useContext(AuthContext);

return (

<nav className={classes.nav}>

<ul>

{ctx.isLoggedIn && (

<li>

<a href='/'>Users</a>

</li>

)}

{ctx.isLoggedIn && (

<li>

<a href='/'>Admin</a>

</li>

)}

{ctx.isLoggedIn && (

<li>

<button onClick={ctx.loggedOut}>Logout</button>

</li>

)}

</ul>

</nav>

);

};

export default Navigation;

login.js

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

import Card from '../UI/Card/Card';

import classes from './Login.module.css';

import Button from '../UI/Button/Button';

import AuthContext from '../../store/auth-context';

const emailReducer = (prevState, action) => {

if (action.type === 'USER\_INPUT') {

return { value: action.val, isValid: action.val.includes('@') };

}

if (action.type === 'INPUT\_BLUR') {

return { value: prevState.value, isValid: prevState.value.includes('@') };

}

return { value: '', isValid: false };

};

const passwordReducer = (preState, action) => {

if (action.type === 'USER\_INPUT') {

return {

value: action.val,

isValid: action.val.trim().length > 6,

};

}

if (action.type === 'INPUT\_BLUR') {

return {

value: preState.value,

isValid: preState.value.trim().length > 6,

};

}

return { value: '', isValid: false };

};

const Login = (props) => {

const ctx = useContext(AuthContext);

// const [enteredEmail, setEnteredEmail] = useState('');

// const [emailIsValid, setEmailIsValid] = useState();

// const [enteredPassword, setEnteredPassword] = useState('');

// const [passwordIsValid, setPasswordIsValid] = useState();

const [formIsValid, setFormIsValid] = useState(false);

const [emailState, dispatchEmail] = useReducer(emailReducer, {

value: '',

isValid: null,

});

const [passwordState, dispatchPassword] = useReducer(passwordReducer, {

value: '',

isValid: null,

});

useEffect(() => {

const identifier = setTimeout(() => {

console.log('Login.js check form is valid');

setFormIsValid(emailState.isValid && passwordState.isValid);

}, 500);

return () => {

console.log('Cleanup');

clearTimeout(identifier);

};

}, [emailState.isValid, passwordState.isValid]);

const emailChangeHandler = (event) => {

dispatchEmail({

type: 'USER\_INPUT',

val: event.target.value,

});

};

const passwordChangeHandler = (event) => {

//setEnteredPassword(event.target.value);

dispatchPassword({

type: 'USER\_INPUT',

val: event.target.value,

});

};

const validateEmailHandler = () => {

// setEmailIsValid(enteredEmail.includes('@'));

dispatchEmail({

type: 'INPUT\_BLUR',

});

};

const validatePasswordHandler = () => {

//setPasswordIsValid(enteredPassword.trim().length > 6);

dispatchPassword({

type: 'INPUT\_BLUR',

});

};

const submitHandler = (event) => {

event.preventDefault();

ctx.loggedIn(emailState.value, passwordState.value);

// props.onLogin(emailState.value, passwordState.value);

};

return (

<Card className={classes.login}>

<form onSubmit={(event) => submitHandler(event)}>

<div

className={`${classes.control} ${

emailState.isValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='email'>E-Mail</label>

<input

type='email'

id='email'

value={emailState.value}

onChange={emailChangeHandler}

onBlur={validateEmailHandler}

/>

</div>

<div

className={`${classes.control} ${

passwordState.isValid === false ? classes.invalid : ''

}`}

>

<label htmlFor='password'>Password</label>

<input

type='password'

id='password'

value={passwordState.value}

onChange={passwordChangeHandler}

onBlur={validatePasswordHandler}

/>

</div>

<div className={classes.actions}>

<Button type='submit' className={classes.btn} disabled={!formIsValid}>

Login

</Button>

</div>

</form>

</Card>

);

};

export default Login;

### useImperativeHandle

useImperativeHandle(ref, createHandle, [deps])

useImperativeHandle customizes the instance value that is exposed to parent components when using ref. As always, imperative code using refs should be avoided in most cases. useImperativeHandle should be used with [forwardRef](https://reactjs.org/docs/react-api.html#reactforwardref)

Login.js

import React, {

useState,

useEffect,

useReducer,

useContext,

useRef,

} from 'react';

import Card from '../UI/Card/Card';

import classes from './Login.module.css';

import Button from '../UI/Button/Button';

import AuthContext from '../../store/auth-context';

import Input from '../UI/Input/Input';

const emailReducer = (state, action) => {

if (action.type === 'USER\_INPUT') {

return { value: action.val, isValid: action.val.includes('@') };

}

if (action.type === 'INPUT\_BLUR') {

return { value: state.value, isValid: state.value.includes('@') };

}

return { value: '', isValid: false };

};

const passwordReducer = (state, action) => {

if (action.type === 'USER\_INPUT') {

return { value: action.val, isValid: action.val.trim().length > 6 };

}

if (action.type === 'INPUT\_BLUR') {

return { value: state.value, isValid: state.value.trim().length > 6 };

}

return { value: '', isValid: false };

};

const Login = (props) => {

const [formIsValid, setFormIsValid] = useState(false);

const [emailState, dispatchEmail] = useReducer(emailReducer, {

value: '',

isValid: null,

});

const [passwordState, dispatchPassword] = useReducer(passwordReducer, {

value: '',

isValid: null,

});

const authCtx = useContext(AuthContext);

useEffect(() => {

console.log('EFFECT RUNNING');

return () => {

console.log('EFFECT CLEANUP');

};

}, []);

const { isValid: emailIsValid } = emailState;

const { isValid: passwordIsValid } = passwordState;

const emailInputRef = useRef();

const passwordInputrRef = useRef();

useEffect(() => {

const identifier = setTimeout(() => {

console.log('Checking form validity!');

setFormIsValid(emailIsValid && passwordIsValid);

}, 500);

return () => {

console.log('CLEANUP');

clearTimeout(identifier);

};

}, [emailIsValid, passwordIsValid]);

const emailChangeHandler = (event) => {

dispatchEmail({ type: 'USER\_INPUT', val: event.target.value });

};

const passwordChangeHandler = (event) => {

dispatchPassword({ type: 'USER\_INPUT', val: event.target.value });

};

const validateEmailHandler = () => {

dispatchEmail({ type: 'INPUT\_BLUR' });

};

const validatePasswordHandler = () => {

dispatchPassword({ type: 'INPUT\_BLUR' });

};

const submitHandler = (event) => {

event.preventDefault();

if (formIsValid) {

authCtx.onLogin(emailState.value, passwordState.value);

} else {

if (!emailIsValid) {

emailInputRef.current.focus(); //calling function of Input component

} else if (!passwordIsValid) {

passwordInputrRef.current.focus();

}

}

};

return (

<Card className={classes.login}>

<form onSubmit={submitHandler}>

<Input

ref={emailInputRef} //pass the ref

isValid={emailIsValid}

id='email'

onChange={emailChangeHandler}

onBlur={validateEmailHandler}

value={emailState.value}

type='email'

label='E-mail'

/>

<Input

ref={passwordInputrRef}

isValid={passwordIsValid}

id='password'

onChange={passwordChangeHandler}

onBlur={validatePasswordHandler}

value={passwordState.value}

type='password'

label='Password'

/>

<div className={classes.actions}>

<Button type='submit' className={classes.btn}>

Login

</Button>

</div>

</form>

</Card>

);

};

export default Login;

Input.js

import React, { useRef, useImperativeHandle } from 'react';

import classes from './Input.module.css';

//to accept ref from parent use forwardRef

const Input = React.forwardRef((props, ref) => {

const inputRef = useRef();

//call this functoin outside

const activate = () => {

inputRef.current.focus();

};

useImperativeHandle(ref, () => {

return {

focus: activate, //all the property to expose outside

};

});

return (

<div

className={`${classes.control} ${

props.isValid === false ? classes.invalid : ''

}`}

>

<label htmlFor={props.id}>{props.label}</label>

<input

ref={inputRef}

type={props.type}

id={props.id}

value={props.value}

onChange={props.onChange}

onBlur={props.onBlur}

/>

</div>

);

});

export default Input;

use of context in application

Inside the store folder

1)create a separate file like cart context where we define all the possible instance and methods

CartContext.js

import React from 'react';

const CartContext = React.createContext({

items: [],

totalAmount: 0,

addItem: (item) => {},

removeItem: (id) => {},

});

export default CartContext;

2) Create provider Component

CartProvider.js It implements all the methods and instance of cartcontext

import React from 'react';

import CartContext from './cart-context';

const CartProvider = (props) => {

const addItemToCartHandler = (item) => {};

const removeItemFromCartHandler = (id) => {};

const cartContext = {

items: [],

totalAmount: 0,

addItem: addItemToCartHandler,

removeItem: removeItemFromCartHandler,

};

return (

<CartContext.Provider value={cartContext}>

{props.children}

</CartContext.Provider>

);

};

export default CartProvider;

3) Wrap the component where we need this context data

app.js

import { useState } from 'react';

import Cart from './components/Cart/Cart';

import Header from './components/Layout/Header';

import Meals from './components/Meals/Meals';

import CartProvider from './store/CartProvider';

function App() {

const [cartIsShown, setCartIsShown] = useState(false);

const showCartHandler = () => {

setCartIsShown(true);

};

const hideCartHandler = () => {

setCartIsShown(false);

};

return (

<CartProvider>

{cartIsShown && <Cart onClose={hideCartHandler} />}

<Header onShowCart={showCartHandler} />

<main>

<Meals />

</main>

</CartProvider>

);

}

export default App;

4) useContext is use to access all context instance variables and methods

HeaderCartButton.js

import { useContext } from 'react';

import CartIcon from '../Cart/CartIcon';

import classes from './HeaderCartButton.module.css';

import CartContext from '../../store/cart-context';

const HeaderCartButton = (props) => {

const cartCtx = useContext(CartContext);

const numberOfCartItems = cartCtx.items.reduce((currNumber, item) => {

return currNumber + item.amount;

}, 0);

return (

<button className={classes.button} onClick={props.onClick}>

<span className={classes.icon}>

<CartIcon />

</span>

<span>Your Cart</span>

<span className={classes.badge}>{numberOfCartItems}</span>

</button>

);

};

export default HeaderCartButton;

**React.memo**  it stop re-evaluating of component

import React from 'react';

import MyParagraph from './MyParagraph';

const DemoOutput = (props) => {

console.log('DemoOutput RUNNING');

return <MyParagraph>{props.show ? 'This is new!' : ''}</MyParagraph>;

};

export default React.memo(DemoOutput);

it is used where we use large number of nested tree component. it compare previous props with new props and then evaluate the react component.

It work on primitive data types

useCallback()

it is hook which store reference value or function in memory to prevent component re-evaluating

app.js

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

import Button from './components/UI/Button/Button';

import DemoOutput from './components/Demo/DemoOutput';

import './App.css';

function App() {

const [showParagraph, setShowParagraph] = useState(false);

console.log('APP RUNNING');

//useCallback

const toggleParagraphHandler = useCallback(() => {

setShowParagraph((prevShowParagraph) => !prevShowParagraph);

}, []);

return (

<div className='app'>

<h1>Hi there!</h1>

<DemoOutput show={false} />

<Button onClick={toggleParagraphHandler}>Toggle Paragraph!</Button>

</div>

);

}

export default App;

Button.js

import React from 'react';

import classes from './Button.module.css';

const Button = (props) => {

console.log('Button RUNNING');

return (

<button

type={props.type || 'button'}

className={`${classes.button} ${props.className}`}

onClick={props.onClick}

disabled={props.disabled}

>

{props.children}

</button>

);

};

export default React.memo(Button);

useMemo is used to memorize the complex and time consuming statement for eg: sorting,

app.js

import React, { useState, useCallback, useMemo } from 'react';

import './App.css';

import DemoList from './components/Demo/DemoList';

import Button from './components/UI/Button/Button';

function App() {

const [listTitle, setListTitle] = useState('My List');

const changeTitleHandler = useCallback(() => {

setListTitle('New Title');

}, []);

//memoiaz array because it is reference type it evaluate each time when Parent component re-evalute and it also re-evalute all including component inside it

const listItems = useMemo(() => [5, 3, 1, 10, 9], []);

return (

<div className='app'>

<DemoList title={listTitle} items={listItems} />

<Button onClick={changeTitleHandler}>Change List Title</Button>

</div>

);

}

export default App;

demolish.js

import React, { useMemo } from 'react';

import classes from './DemoList.module.css';

const DemoList = (props) => {

const { items } = props;

//here memorised because complex computation is evaluated here. eg sorting

const sortedList = useMemo(() => {

console.log('Items sorted');

return items.sort((a, b) => a - b);

}, [items]);

console.log('DemoList RUNNING');

return (

<div className={classes.list}>

<h2>{props.title}</h2>

<ul>

{sortedList.map((item) => (

<li key={item}>{item}</li>

))}

</ul>

</div>

);

};

export default React.memo(DemoList);