**Assignment No:-74**

Name:-Suryawanshi Sangramsingh Sambhaji

Batch: - Delta - DCA (Java) 2024 Date:-11/9/2024

1. Toggle Visibility of an Element

Create a React component with a button and a paragraph of text. When the button is clicked, the paragraph should toggle between being visible and hidden.

Hint: Use useState to manage the visibility state.

import {useState} from 'react';

const Toggle =()=>{

const[isVisible,SetIsVisble] = useState(true);

function update()

{

SetIsVisble(!isVisible);

}

return <>

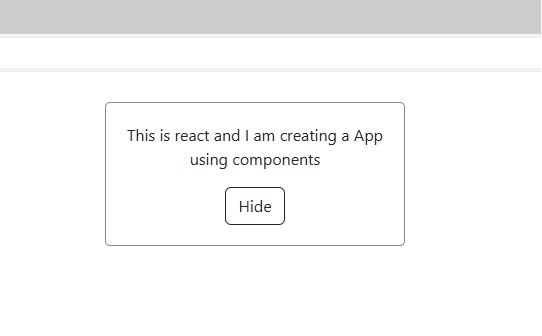
{isVisible && <p>This is react and I am creating a App using components</p>}

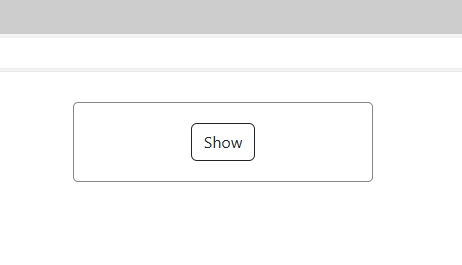
<button type="button" className="btn active" data-bs-toggle="button" aria-pressed="true" onClick={update} >{isVisible ? 'Hide' : 'Show'}</button>

</>

}

export default Toggle;





2. Counter with Reset Functionality

Build a simple counter component that increments the count by 1 each time a button is clicked. Add a second button that resets the count back to zero.

Hint: You will need two onClick functions—one for incrementing the count and one for resetting.

import {useState} from 'react';

export default function Counter()

{

const[count,setCount]=useState(0);

return <>

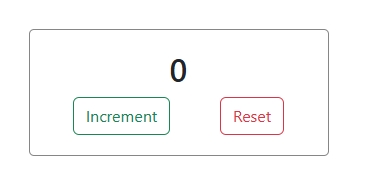
<h1>{count}</h1>

<button type="button" className="btn btn-outline-success" style={{marginRight:"50px"}} onClick={()=>setCount(count+1)} >Increment</button>

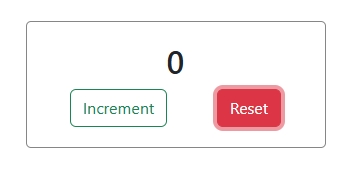
<button type="button" className="btn btn-outline-danger" onClick={()=>setCount(0)}>Reset</button>

</>

}







3. Input Field and Dynamic Greeting

Create a React component with an input field. As the user types their name into the input, dynamically update a greeting message (e.g., "Hello, [name]!").

Hint: Use onChange to capture input and update the state.

import { useState } from 'react';

const Greet = () => {

const [name, setName] = useState('');

function update(e) {

setName(e.target.value);

}

return <>

<h1>Hello, {name} !</h1>

<div class="input-group input-group-sm mb-3">

<input type="text" class="form-control" aria-label="Sizing example input" aria-describedby="inputGroup-sizing-sm" onChange={update} />

</div>

</>

}

export default Greet;



4. Color Changing Box

Create a box (e.g., div with a fixed width and height) and a button. Each time the button is clicked, the box's background color should change randomly.

Hint: Use inline styles and generate a random color using JavaScript.

import { useState } from 'react';

const Color = () => {

const [color, setColor] = useState('#ffffff');

function getRandomColor() {

const letters = '0123456789ABCDEF';

let color = '#';

for (let i = 0; i < 6; i++) {

color += letters[Math.floor(Math.random() \* 16)];

}

return color;

}

function chageColor() {

setColor(getRandomColor());

}

return <>

<div style={

{

width: '200px',

height: '200px',

backgroundColor: color,

border: '1px solid #000',

padding: '20px',

margin: 'auto',

marginBottom: '20px'

}

}>

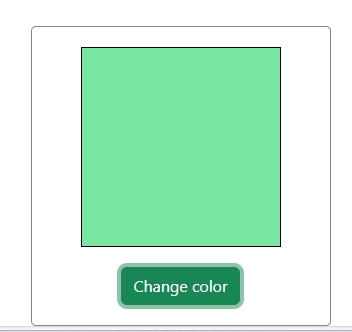
</div>

<button type="button" onClick={chageColor} className="btn btn-outline-success">Change color</button>

</>

}

export default Color;



5. Form Validation

Create a form with an email and password input field. When the form is submitted, validate that the email is in a valid format and the password is at least 8 characters long. Display an error message if validation fails.

Hint: Use the form’s onSubmit event and useState to handle input values and error messages.

import { useState } from "react";

export default function FormValidation() {

const [passError, setPassError] = useState('');

const [mailerror, setMailError] = useState('');

function validateEmail(e) {

const mail = e.target.value;

setMailError('');

let cnt = 0;

for (let i = 0; i < mail.length; i++) {

if (mail[i] === '@') {

cnt++;

}

}

if (cnt === 0) {

setMailError('Mail id should contain @ symbol.')

}

}

function validatePassword(e) {

const pass = e.target.value;

setPassError('')

if (pass.length < 8) {

setPassError('Password must be 8 letters or More')

}

}

return <>

<form>

<div class="form-group">

<label for="exampleInputEmail1">Email address</label>

<p style={{ color: 'red' }}>{mailerror}</p>

<input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email" onChange={validateEmail} />

<small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone.</small>

</div>

<div class="form-group">

<label for="exampleInputPassword1">Password</label>

<p style={{ color: 'red' }}>{passError}</p>

<input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password" onChange={validatePassword} />

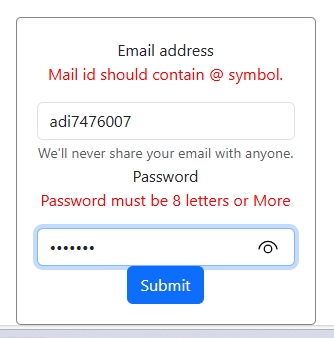
</div>

<button type="submit" class="btn btn-primary">Submit</button>

</form>

</>

}



6. List Rendering and Item Removal

Create a React component that renders a list of items. Each item should have a "Remove" button next to it. When the button is clicked, remove the item from the list.

Hint: Use useState to manage the list and filter() to remove the selected item.

7. Disable Button After Click

Create a button that, once clicked, becomes disabled and cannot be clicked again.

Hint: Use the disabled attribute in combination with state to control the button's clickable status.

import { useState } from "react";

export default function DisableBtn() {

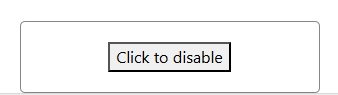
const [isdisable, setIsDisable] = useState(false);

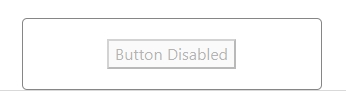
return <>

{isdisable ? <button disabled>Button Disabled</button> : <button onClick={() => setIsDisable(true)}>Click to disable</button>}

</>

}





8. Temperature Converter

Create two input fields: one for Celsius and one for Fahrenheit. As the user types into either field, automatically convert the value to the other temperature scale.

Hint: Use two pieces of state for the temperature values and implement the conversion formulas.

import {useState} from 'react';

export default function Convertor()

{

const [celcius, setCelcuis] = useState("");

const [fahrenhite, setFahrenhite] = useState("");

function handleCelcius(e)

{

const val = e.target.value;

setCelcuis(val);

if(val === ""){

setFahrenhite("");

}

else

{

setFahrenhite(((val\*9)/5+32).toFixed(2));

}

}

function handleFahrenhite(e)

{

const val = e.target.value;

setFahrenhite(val);

if(val === ""){

setCelcuis("");

}

else

{

setCelcuis(((val - 32) \* 5 / 9).toFixed(2));

}

}

return(

<div>

<label> Celcius : </label><input type='number' value={celcius} onChange={handleCelcius}></input>

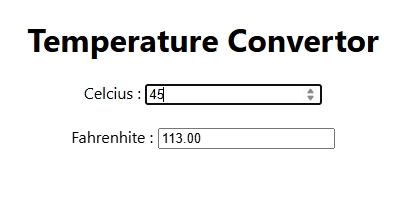
<br/><br/>

<label> Fahrenhite : </label><input type='number' value={fahrenhite} onChange={handleFahrenhite}></input>

</div>

)

}



9. Increment and Decrement Buttons

Create a counter component with two buttons: one to increment the counter and one to decrement it. Ensure that the counter value cannot go below zero.

Hint: Use an if condition to prevent the counter from going negative.

import { useState } from 'react';

export default function CounterV2() {

const [count, setCount] = useState(0);

function check() {

if (count > 0) {

setCount(count - 1)

}

else {

setCount(0);

}

}

return <>

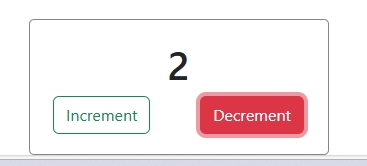
<h1>{count}</h1>

<button type="button" className="btn btn-outline-success" style={{ marginRight: "50px" }} onClick={() => setCount(count + 1)} >Increment</button>

<button type="button" className="btn btn-outline-danger" onClick={check}>Decrement</button>

</>

}



10. Shopping Cart Item Quantity Update

Create a small shopping cart system where each product has a quantity. Use "+" and "-" buttons next to each product to increase or decrease the quantity. If the quantity goes to zero, display a message like "Out of stock."

Hint: Use useState for managing each product’s quantity and conditionally render the out-of-stock message