**Q.1 Add a button that disables itself after 3 clicks** (use state to track clicks). ......................give me react code according to necessary js files

🔹 1. index.js

jsx

CopyEdit

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

—

🔹 2. App.js

jsx

CopyEdit

import React from 'react';

import ClickLimiter from './components/ClickLimiter';

function App() {

return (

<div>

<h1>Click Limiter Example</h1>

<ClickLimiter />

</div>

);

}

export default App;

—

🔹 3. components/ClickLimiter.js

jsx

CopyEdit

import React, { useState } from 'react';

function ClickLimiter() {

const [clicks, setClicks] = useState(0);

const [isDisabled, setIsDisabled] = useState(false);

const handleClick = () => {

const newCount = clicks + 1;

setClicks(newCount);

if (newCount >= 3) {

setIsDisabled(true);

}

};

return (

<div>

<button onClick={handleClick} disabled={isDisabled}>

{isDisabled ? 'Disabled' : `Clicked ${clicks} time(s)`}

</button>

</div>

);

}

export default ClickLimiter;

**2.** **Implement a <select> dropdown that updates state with the selected value**........,give me code according to necessary js files

**ChatGPT said:**

Here’s the complete React.js code for your task:

✅ **Implement a <select> dropdown that updates state with the selected value.**

**🔹 Code Files**

**1. index.js**

🔸 Main entry point for the React app.

jsx

CopyEdit

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**2. App.js**

🔸 Loads the DropdownSelect component.

jsx

CopyEdit

import React from 'react';

import DropdownSelect from './components/DropdownSelect';

function App() {

return (

<div>

<h1>Dropdown Select Example</h1>

<DropdownSelect />

</div>

);

}

export default App;

**3. components/DropdownSelect.js**

🔸 A <select> dropdown that updates state with the selected value.

jsx

CopyEdit

import React, { useState } from 'react';

function DropdownSelect() {

const [selectedValue, setSelectedValue] = useState(''); // initial state is empty

const handleChange = (e) => {

setSelectedValue(e.target.value); // update state with selected value

};

return (

<div>

<h2>Select a Fruit:</h2>

<select value={selectedValue} onChange={handleChange} style={{ padding: '8px', fontSize: '16px' }}>

<option value="">--Please choose an option--</option>

<option value="Apple">Apple</option>

<option value="Banana">Banana</option>

<option value="Orange">Orange</option>

<option value="Grapes">Grapes</option>

<option value="Mango">Mango</option>

</select>

{selectedValue && <p>You selected: {selectedValue}</p>}

</div>

);

}

export default DropdownSelect;

**3. Build a AADHAR Registration form with an input field and a submit button..**...........,give me code according to necessary js files

**ChatGPT said:**

Here’s the complete React.js code for building an **AADHAR Registration Form** with an input field and a submit button:

✅ **Build an AADHAR Registration form with an input field and a submit button.**

**🔹 Code Files**

**1. index.js**

🔸 Main entry point for the React app.

jsx

CopyEdit

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**2. App.js**

🔸 Loads the AadharForm component.

jsx

CopyEdit

import React from 'react';

import AadharForm from './components/AadharForm';

function App() {

return (

<div>

<h1>AADHAR Registration Form</h1>

<AadharForm />

</div>

);

}

export default App;

**3. components/AadharForm.js**

🔸 A simple AADHAR Registration form with an input field and a submit button.

jsx

CopyEdit

import React, { useState } from 'react';

function AadharForm() {

const [aadharNumber, setAadharNumber] = useState('');

const handleChange = (e) => {

setAadharNumber(e.target.value);

};

const handleSubmit = (e) => {

e.preventDefault();

// Validate the Aadhar number (e.g., 12 digits)

if (aadharNumber.length === 12) {

alert(`AADHAR Number: ${aadharNumber} Registered Successfully!`);

} else {

alert('Please enter a valid 12-digit AADHAR number.');

}

};

return (

<form onSubmit={handleSubmit} style={{ maxWidth: '400px', margin: '0 auto', padding: '20px', border: '1px solid #ccc' }}>

<div style={{ marginBottom: '10px' }}>

<label htmlFor="aadharNumber" style={{ display: 'block', marginBottom: '5px' }}>Enter AADHAR Number:</label>

<input

type="text"

id="aadharNumber"

value={aadharNumber}

onChange={handleChange}

placeholder="Enter 12-digit AADHAR number"

maxLength="12"

style={{ padding: '8px', width: '100%' }}

/>

</div>

<button type="submit" style={{ padding: '10px 20px', backgroundColor: '#4CAF50', color: 'white', border: 'none', cursor: 'pointer' }}>

Submit

</button>

</form>

);

}

export default AadharForm;

**4. Add Form Validation in above program**..........give me code according to necessary js files

**🔹 Updated Code Files**

**1. index.js**

🔸 Main entry point for the React app.

jsx

CopyEdit

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**2. App.js**

🔸 Loads the AadharForm component.

jsx

CopyEdit

import React from 'react';

import AadharForm from './components/AadharForm';

function App() {

return (

<div>

<h1>AADHAR Registration Form</h1>

<AadharForm />

</div>

);

}

export default App;

**3. components/AadharForm.js**

🔸 **AADHAR Registration form with added validation:**

* Checks if the Aadhar number is exactly 12 digits.
* Validates that only numeric characters are allowed.

jsx

CopyEdit

import React, { useState } from 'react';

function AadharForm() {

const [aadharNumber, setAadharNumber] = useState('');

const [error, setError] = useState('');

const handleChange = (e) => {

const value = e.target.value;

// Allow only numeric input

if (/^\d\*$/.test(value)) {

setAadharNumber(value);

setError('');

} else {

setError('Please enter only numeric digits.');

}

};

const handleSubmit = (e) => {

e.preventDefault();

// Validation for 12 digits

if (aadharNumber.length === 12) {

alert(`AADHAR Number: ${aadharNumber} Registered Successfully!`);

} else {

setError('Please enter a valid 12-digit AADHAR number.');

}

};

return (

<form onSubmit={handleSubmit} style={{ maxWidth: '400px', margin: '0 auto', padding: '20px', border: '1px solid #ccc' }}>

<div style={{ marginBottom: '10px' }}>

<label htmlFor="aadharNumber" style={{ display: 'block', marginBottom: '5px' }}>Enter AADHAR Number:</label>

<input

type="text"

id="aadharNumber"

value={aadharNumber}

onChange={handleChange}

placeholder="Enter 12-digit AADHAR number"

maxLength="12"

style={{ padding: '8px', width: '100%' }}

/>

</div>

{error && <p style={{ color: 'red', fontSize: '14px' }}>{error}</p>}

<button type="submit" style={{ padding: '10px 20px', backgroundColor: '#4CAF50', color: 'white', border: 'none', cursor: 'pointer' }}>

Submit

</button>

</form>

);

}

export default AadharForm;

**Key Updates:**

* **State for Error (error)**: The error state is used to display error messages.
* **Validation for Numeric Input**:
  + The handleChange function now allows only **numeric digits** (/^\d\*$/).
  + If a non-numeric character is typed, it displays an error: "Please enter only numeric digits."
* **Validation for 12 Digits**:
  + The handleSubmit function now checks if the Aadhar number is exactly **12 digits long**.
  + If the number is not 12 digits, it shows an error message: "Please enter a valid 12-digit AADHAR number."
* **Error Display**: If there’s any error, it will be displayed below the input field in **red text**.

**How It Works:**

* The form now validates both:
  1. **Only numeric input** is allowed.
  2. The **Aadhar number must be exactly 12 digits** long.
* Any errors in the input will be shown dynamically under the form.

Let me know if you'd like to further customize:

* More fields for the form (e.g., name, address).
* Error messages for specific input fields.
* Styling enhancements or custom themes.

**5. Write a program to demonstrate functional component life cycle phases.........give me code according to necessary js files**

**ChatGPT said:**

In React, **functional components** do not have the traditional lifecycle methods like class components. However, React provides **hooks** such as useEffect to manage side effects and mimic the behavior of lifecycle methods in functional components.

Here’s a simple demonstration of **functional component lifecycle phases** using the useEffect hook.

**Lifecycle Phases:**

* **Mounting**: When the component is created and added to the DOM (similar to componentDidMount in class components).
* **Updating**: When the component updates due to changes in state or props (similar to componentDidUpdate in class components).
* **Unmounting**: When the component is removed from the DOM (similar to componentWillUnmount in class components).

**🔹 Code Files**

**1. index.js**

🔸 Main entry point for the React app.

jsx

CopyEdit

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**2. App.js**

🔸 Loads the LifecycleDemo component.

jsx

CopyEdit

import React from 'react';

import LifecycleDemo from './components/LifecycleDemo';

function App() {

return (

<div>

<h1>Functional Component Lifecycle Demo</h1>

<LifecycleDemo />

</div>

);

}

export default App;

**3. components/LifecycleDemo.js**

🔸 Demonstrates the lifecycle phases in a functional component using useEffect.

jsx

CopyEdit

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

function LifecycleDemo() {

const [counter, setCounter] = useState(0);

// Mounting Phase (componentDidMount)

useEffect(() => {

console.log('Component mounted');

alert('Component has mounted');

// Cleanup Phase (componentWillUnmount)

return () => {

console.log('Component will unmount');

alert('Component will unmount');

};

}, []); // Empty dependency array means it only runs once when the component mounts

// Updating Phase (componentDidUpdate)

useEffect(() => {

if (counter !== 0) {

console.log(`Component updated: counter is now ${counter}`);

alert(`Counter updated to ${counter}`);

}

}, [counter]); // Runs when `counter` state changes

const incrementCounter = () => {

setCounter(counter + 1);

};

const decrementCounter = () => {

setCounter(counter - 1);

};

return (

<div>

<h2>Counter: {counter}</h2>

<button onClick={incrementCounter}>Increment</button>

<button onClick={decrementCounter}>Decrement</button>

</div>

);

}

export default LifecycleDemo;

**6. Create a timer that updates the seconds elapsed every seconds...............give me react code according to necessary js files**

**ChatGPT said:**

Here is a simple **timer** in React that updates the seconds elapsed every second:

**🔹 Code Files**

**1. index.js**

🔸 Main entry point for the React app.

jsx

CopyEdit

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**2. App.js**

🔸 Loads the Timer component.

jsx

CopyEdit

import React from 'react';

import Timer from './components/Timer';

function App() {

return (

<div>

<h1>Timer Example</h1>

<Timer />

</div>

);

}

export default App;

**3. components/Timer.js**

🔸 A **timer component** that updates every second.

jsx

CopyEdit

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

function Timer() {

const [seconds, setSeconds] = useState(0); // State to keep track of the seconds

// useEffect to update seconds every second

useEffect(() => {

const intervalId = setInterval(() => {

setSeconds(prevSeconds => prevSeconds + 1); // Increment the seconds

}, 1000);

// Cleanup function to clear the interval when the component is unmounted

return () => clearInterval(intervalId);

}, []); // Empty dependency array to only run once when the component mounts

return (

<div>

<h2>Seconds Elapsed: {seconds}</h2>

</div>

);

}

export default Timer;

**You said:**

**7. Use useState to manage the visibility of a paragraph (toggle text with a button)............**give me react code according to necessary js files

**🔹 Code Files**

**1. index.js**

🔸 Main entry point for the React app.

jsx

CopyEdit

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

**2. App.js**

🔸 Loads the VisibilityToggle component.

jsx

CopyEdit

import React from 'react';

import VisibilityToggle from './components/VisibilityToggle';

function App() {

return (

<div>

<h1>Toggle Paragraph Visibility</h1>

<VisibilityToggle />

</div>

);

}

export default App;

**3. components/VisibilityToggle.js**

🔸 **Visibility toggle** component that shows and hides the paragraph when the button is clicked.

jsx

CopyEdit

import React, { useState } from 'react';

function VisibilityToggle() {

const [isVisible, setIsVisible] = useState(false); // State to manage the visibility

const toggleVisibility = () => {

setIsVisible(prevState => !prevState); // Toggle the visibility state

};

return (

<div>

<button onClick={toggleVisibility} style={{ padding: '10px 20px', marginBottom: '20px' }}>

{isVisible ? 'Hide' : 'Show'} Paragraph

</button>

{isVisible && (

<p>

This is a paragraph that can be toggled visible or hidden using the button above.

</p>

)}

</div>

);

}

export default VisibilityToggle;