**React ETE-Questions**

**Q1. a)** **Write a react code to build a simple search filter functionality to display a filtered list on the search query entered by the user.**

**Sol.**

searchFilter.js

import React, { useState } from "react";

const SearchFilter = ({ data }) => {

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

const filteredData = data.filter((item) =>

item.toLowerCase().includes(query.toLowerCase())

);

return (

<div>

<input type="text" value={query} onChange={(e) => setQuery(e.target.value)} />

<ul>

{filteredData.map((item, index) => (

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

))}

</ul>

</div>

);

};

export default SearchFilter;

App.js

import React from "react";

import SearchFilter from "./SearchFilter";

const data = ["apple", "banana", "orange", "pear", "pineapple"];

const App = () => {

return (

<div>

<SearchFilter data={data} />

</div>

);

};

export default App;

**Q1. b) Create a simple counter using react which increments or decrements count dynamically on screen as the user clicks on the button.**

**Sol.**

import React, { useState } from "react";

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

const incrementCount = () => { setCount(count + 1);

};

const decrementCount = () => { setCount(count - 1);

};

return (

<div>

<h1>Count: {count}</h1>

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

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

</div>

);

}

export default App;

**Q2. a) Write a react code to print each item from the list on the page using Array map() function to display each item on the page.**

**Sol.**

list.js

import React from "react";

const List = ({ items }) => {

return (

<ul>

{items.map((item, index) => (

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

))}

</ul>

);

};

export default List;

App.js

import React from "react";

import List from "./List";

const items = ["apple", "banana", "orange", "pear", "pineapple"];

const App = () => {

return (

<div>

<List items={items} />

</div>

);

};

export default App;

**Q2. b) Create an accordian that toggles text content on click of the accordian header using react state.**

**Sol.**

accordian.js

import React, { useState } from "react";

const Accordion = ({ title, content }) => {

const [isOpen, setIsOpen] = useState(false);

const toggleAccordion = () => {

setIsOpen(!isOpen);

};

return (

<div className="accordion">

<div className="accordion-header" onClick={toggleAccordion}>

{title}

{isOpen ? <i className="fas fa-chevron-up"></i> : <i className="fas fa-chevron-down"></i>}

</div>

{isOpen && <div className="accordion-content">{content}</div>}

</div>

);

};

export default Accordion;

App.js

import React from "react";

import Accordion from "./Accordion";

const App = () => {

return (

<div>

<Accordion title="Accordion Title" content="This is the content of the accordion." />

</div>

);

};

export default App;

**Q3. a) Write a react code to create an image slide, where users can view multiple images with next/previous buttons.**

**Sol.**

imageSlider.js

import React, { useState } from "react";

const ImageSlider = ({ images }) => {

const [currentImageIndex, setCurrentImageIndex] = useState(0);

const goToPreviousImage = () => {

if (currentImageIndex === 0) {

setCurrentImageIndex(images.length - 1);

} else {

setCurrentImageIndex(currentImageIndex - 1);

}

};

const goToNextImage = () => {

if (currentImageIndex === images.length - 1) {

setCurrentImageIndex(0);

} else {

setCurrentImageIndex(currentImageIndex + 1);

}

};

return (

<div className="image-slider">

<button onClick={goToPreviousImage}>Previous</button>

<img src={images[currentImageIndex]} alt="Slide" />

<button onClick={goToNextImage}>Next</button>

</div>

);

};

export default ImageSlider;

App.js

import React from "react";

import ImageSlider from "./ImageSlider";

const images = ["https://example.com/image1.jpg", "https://example.com/image2.jpg", "https://example.com/image3.jpg"];

const App = () => {

return (

<div>

<ImageSlider images={images} />

</div>

);

};

export default App;

**Q3. b) Write a react code to display a checklist with multiple options that can select and the selected options are dynamically displayed on the screen.**

**Sol.**

checklist.js

import React, { useState } from "react";

const Checklist = () => {

const [options, setOptions] = useState([

{ id: 1, label: "Option 1", checked: false },

{ id: 2, label: "Option 2", checked: false },

{ id: 3, label: "Option 3", checked: false },

{ id: 4, label: "Option 4", checked: false }

]);

const handleOptionChange = (optionId) => {

const updatedOptions = options.map((option) => {

if (option.id === optionId) {

return { ...option, checked: !option.checked };

} else {

return option;

}

});

setOptions(updatedOptions);

};

const selectedOptions = options.filter((option) => option.checked);

return (

<div>

{options.map((option) => (

<label key={option.id}>

<input

type="checkbox"

checked={option.checked}

onChange={() => handleOptionChange(option.id)}

/>

{option.label}

</label>

))}

<div>

<h3>Selected Options:</h3>

<ul>

{selectedOptions.map((option) => (

<li key={option.id}>{option.label}</li>

))}

</ul>

</div>

</div>

);

};

export default Checklist;

App.js

import React from "react";

import Checklist from "./Checklist";

const App = () => {

return (

<div>

<Checklist />

</div>

);

};

export default App;

**Q4. a) Write a react code for simple login form where the user login by entering their username and password. The form inputs are validated to check if correct information is entered and the error messages are validation fails. The login form is hidden and the “Welcome, ${name}” message is shown when the user login is successful.**

**Sol.**

loginForm.js

import React, { useState } from 'react';

function LoginForm() {

const [username, setUsername] = useState('');

const [password, setPassword] = useState('');

const [loggedIn, setLoggedIn] = useState(false);

const [errorMessage, setErrorMessage] = useState('');

function handleSubmit(e) {

e.preventDefault();

if (username === 'exampleuser' && password === 'password123') {

setLoggedIn(true);

} else {

setErrorMessage('Invalid username or password');

}

}

function handleUsernameChange(e) {

setUsername(e.target.value);

}

function handlePasswordChange(e) {

setPassword(e.target.value);

}

if (loggedIn) {

return <h1>Welcome, {username}!</h1>;

}

return (

<form onSubmit={handleSubmit}>

<div>

<label htmlFor="username">Username:</label>

<input

type="text"

id="username"

value={username}

onChange={handleUsernameChange}

required

/>

</div>

<div>

<label htmlFor="password">Password:</label>

<input

type="password"

id="password"

value={password}

onChange={handlePasswordChange}

required

/>

</div>

{errorMessage && <div>{errorMessage}</div>}

<button type="submit">Log in</button>

</form>

);

}

export default LoginForm;

App.js

import React from 'react';

import LoginForm from './LoginForm';

function App() {

return (

<div>

<h1>Login Page</h1>

<LoginForm />

</div>

);

}

export default App;

**Q4. b) Write a React code to collect data from rest API using fetch() in JavaScript combined with useeffect() to load the content on page render.**

**Sol.**

App.js

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

function App() {

const [data, setData] = useState([]);

useEffect(() => {

fetch('https://jsonplaceholder.typicode.com/todos')

.then(response => response.json())

.then(data => setData(data))

.catch(error => console.log(error));

}, []);

return (

<div>

<h1>Todo List</h1>

<ul>

{data.map(item => (

<li key={item.id}>{item.title}</li>

))}

</ul>

</div>

);

}

export default App;

**Q5. a) Write a react code to develop a multi-page application with navigation for home, about and blog pages. The route-based component rendering is implemented using the “react-dom” npm package to allow users to navigate to different pages and render the component with respect to the route.**

**Sol.**

App.js

import React from 'react';

import { BrowserRouter ,Route,Routes } from 'react-router-dom';

import Home from './components/Home';

import About from './components/About';

import Blog from './components/Blog';

import Header from './components/Header';

function App() {

  return (

    <>

<BrowserRouter>

<Header/>

<Routes>

<Route exact path="/" element={<Home/>} />

        <Route path="/about" element={<About/>} />

        <Route path="/blog" element={<Blog/>} />

</Routes>

</BrowserRouter>

    </>

  );

}

export default App;

Header.js

import React from 'react'

import {Link} from 'react-router-dom'

const Header = () => {

return (

<>

<nav>

<ul>

<li>

<Link to="/">Home</Link>

</li>

<li>

<Link to="/about">About</Link>

</li>

<li>

<Link to="/blog">Blog</Link>

</li>

</ul>

</nav>

</>

)

}

export default Header

**Q5. b) Create a context that allows values to be passed from multiple levels of child components without using props.**

**Sol.**

**Q6. a) Create a react code to add 2 numbers. Insert two text boxes and a button inserting values in text boxes User should be able to add 2 numbers with the given button.**

**Sol.**

App.js

import React from 'react';

import AddNumbers from './AddNumbers';

function App() {

return (

<div className="App">

<AddNumbers />

</div>

);

}

export default App;

AddNumbers.js

import React, { useState } from 'react';

function AddNumbers() {

const [num1, setNum1] = useState("");

const [num2, setNum2] = useState("");

const [result, setResult] = useState(0);

function handleAdd() {

const sum = parseInt(num1) + parseInt(num2);

setResult(sum);

}

return (

<div>

<input type="text" value={num1} onChange={e => setNum1(e.target.value)} />

<input type="text" value={num2} onChange={e => setNum2(e.target.value)} />

<button onClick={handleAdd}>Add</button>

<p>Result: {result}</p>

</div>

);

}

export default AddNumbers;

**Q6. b) Create a react code to disable a button using the concept of controlled components.**

**Sol.**

App.js

import React from 'react';

import DisableButton from './DisableButton';

function App() {

return (

<div className="App">

<DisableButton />

</div>

);

}

export default App;

DisableButton.js

import React, { useState } from 'react';

function DisableButton() {

const [disabled, setDisabled] = useState(true);

function handleChange(event) {

setDisabled(!event.target.checked);

}

return (

<div>

<label htmlFor="checkbox">Disable Button</label>

<input type="checkbox" id="checkbox" checked={!disabled} onChange={handleChange} />

<button disabled={disabled}>Click Me</button>

</div>

);

}

export default DisableButton;

**Q7. a) Create a react code to hide element on the screen with the click of a button.**

**Sol.**

App.js

import React from 'react';

import HideElement from './HideElement ';

function App() {

return (

<div className="App">

<HideElement />

</div>

);

}

export default App;

HideElement.js

import React, { useState } from 'react';

function HideElement() {

const [visible, setVisible] = useState(true);

function handleClick() {

setVisible(false);

}

return (

<div>

{visible && <p>Hello, world!</p>}

<button onClick={handleClick}>Hide</button>

</div>

);

}

export default HideElement;

**Q7. b) Create a calculator that can perform basic arithmetic operations additions attraction multiplication and division.**

**Sol.**

**Q8. a) Create a form that takes in a name and email address and displays the entire data below the form.**

**Sol.**

App.js

import React from 'react';

import Form from './Form ';

function App() {

return (

<div className="App">

<DisableButton />

</div>

);

}

export default App;

Form.js

import React, { useState } from 'react';

function Form() {

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

const [email, setEmail] = useState('');

const [data, setData] = useState(null);

function handleSubmit(e) {

e.preventDefault();

setData({ name, email });

setName('');

setEmail('');

}

return (

<div>

<form onSubmit={handleSubmit}>

<label>

Name:

<input

type="text"

value={name}

onChange={(e) => setName(e.target.value)}

/>

</label>

<br />

<label>

Email:

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

/>

</label>

<br />

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

</form>

{data && (

<div>

<p>Name: {data.name}</p>

<p>Email: {data.email}</p>

</div>

)}

</div>

);

}

export default Form;

**Q8. b) Create a dashboard that visualizes data from an API You may consider a fake rest API.**

**Sol.**

App.js

import React from 'react';

import Dashboard from './Dashboard ';

function App() {

return (

<div className="App">

<Dashboard />

</div>

);

}

export default App;

Dashboard.js

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

import axios from 'axios';

function Dashboard() {

const [data, setData] = useState([]);

useEffect(() => {

axios.get('https://jsonplaceholder.typicode.com/posts')

.then(response => {

setData(response.data);

})

.catch(error => {

console.log(error);

});

}, []);

return (

<div>

<h1>Dashboard</h1>

{data.map(item => (

<div key={item.id}>

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

<p>{item.body}</p>

</div>

))}

</div>

);

}

export default Dashboard;

**Q9. a) Create a react code to demonstrate componentDIdMount() & componentDIdUpdate.**

**Sol.**

App.js

import React from 'react';

import ExampleComponent from './ExampleComponent ';

function App() {

return (

<div className="App">

<ExampleComponent />

</div>

);

}

export default App;

ExampleComponent.js

import React, { Component } from 'react';

class ExampleComponent extends Component {

constructor(props) {

super(props);

this.state = {

count: 0

};

}

componentDidMount() {

console.log('Component mounted');

}

componentDidUpdate(prevProps, prevState) {

console.log('Component updated');

if (prevState.count !== this.state.count) {

console.log('Count updated:', this.state.count);

}

}

incrementCount = () => {

this.setState({ count: this.state.count + 1 });

}

render() {

return (

<div>

<h1>Component Example</h1>

<p>Count: {this.state.count}</p>

<button onClick={this.incrementCount}>Increment Count</button>

</div>

);

}

}

export default ExampleComponent;

**Q9. b) Demonstrate different type of CSS styling in react.**

**Sol.**

App.js

**Q10. a) Demonstrate form validation in react. Whenever an input control is left blank a message must display on the web page. For example, if you are applying validation on “Name” field, then it must display, “Name is mandatory”.**

**Sol.**

FormValidation.js

import React, { useState } from 'react';

function FormValidation() {

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

const [email, setEmail] = useState('');

const [message, setMessage] = useState('');

const [nameError, setNameError] = useState('');

const [emailError, setEmailError] = useState('');

const [messageError, setMessageError] = useState('');

const handleSubmit = (event) => {

event.preventDefault();

// Check if name is not blank

if (!name) {

setNameError('Name is mandatory');

} else {

setNameError('');

}

// Check if email is valid

if (!email) {

setEmailError('Email is mandatory');

} else if (!/\S+@\S+\.\S+/.test(email)) {

setEmailError('Email is invalid');

} else {

setEmailError('');

}

// Check if message is not blank

if (!message) {

setMessageError('Message is mandatory');

} else {

setMessageError('');

}

// If there are no errors, submit the form

if (!nameError && !emailError && !messageError) {

alert('Form submitted successfully!');

// Do something with the form data, such as sending it to a server

}

};

return (

<form onSubmit={handleSubmit}>

<label>

Name:

<input type="text" value={name} onChange={(e) => setName(e.target.value)} />

{nameError && <span>{nameError}</span>}

</label>

<br />

<label>

Email:

<input type="email" value={email} onChange={(e) => setEmail(e.target.value)} />

{emailError && <span>{emailError}</span>}

</label>

<br />

<label>

Message:

<textarea value={message} onChange={(e) => setMessage(e.target.value)}></textarea>

{messageError && <span>{messageError}</span>}

</label>

<br />

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

</form>

);

}

export default FormValidation;

**Q10. b) Display the following using createElement().**

* **Michael Jackson**
* **The Weekend**
* **Shakira**
* **Britney Spears**

**Sol.**

import React from 'react';

function App() {

const michaelJackson = React.createElement('h1', null, 'Michael Jackson');

const theWeekend = React.createElement('h1', null, 'The Weekend');

const shakira = React.createElement('h1', null, 'Shakira');

const britneySpears = React.createElement('h1', null, 'Britney Spears');

return (

<div>

{michaelJackson}

{theWeekend}

{shakira}

{britneySpears}

</div>

);

}

export default App;

**Q11. Create a todo list app using react js.**

**Sol.**

Todolist.js

import React, { useState } from "react";

const TodoList = () => {

const [todos, setTodos] = useState([]);

const [newTodo, setNewTodo] = useState("");

const handleNewTodoChange = (event) => {

setNewTodo(event.target.value);

};

const handleTodoAdd = (event) => {

event.preventDefault();

if (newTodo !== "") {

setTodos([...todos, newTodo]);

setNewTodo("");

}

};

const handleTodoDelete = (index) => {

const newTodos = [...todos];

newTodos.splice(index, 1);

setTodos(newTodos);

};

return (

<div>

<form onSubmit={handleTodoAdd}>

<input

type="text"

placeholder="Enter a new todo..."

value={newTodo}

onChange={handleNewTodoChange}

/>

<button type="submit">Add Todo</button>

</form>

<ul>

{todos.map((todo, index) => (

<li key={index}>

{todo}

<button onClick={() => handleTodoDelete(index)}>Delete</button>

</li>

))}

</ul>

</div>

);

};

export default TodoList;

App.js

import React from "react";

import TodoList from "./TodoList";

const App = () => {

return (

<div>

<TodoList />

</div>

);

};

export default App;

**Q12. Write a reactjs code to find out whether the given number is an Armstrong number or not.**

**Sol.**

Armstrong.js

import React, { useState } from "react";

const ArmstrongNumberChecker = () => {

const [number, setNumber] = useState("");

const [isArmstrong, setIsArmstrong] = useState(null);

const handleNumberChange = (event) => {

setNumber(event.target.value);

setIsArmstrong(null);

};

const handleNumberSubmit = (event) => {

event.preventDefault();

const digits = number.toString().split("");

const numDigits = digits.length;

let sum = 0;

digits.forEach((digit) => {

sum += Math.pow(parseInt(digit), numDigits);

});

if (sum === parseInt(number)) {

setIsArmstrong(true);

} else {

setIsArmstrong(false);

}

};

return (

<div>

<form onSubmit={handleNumberSubmit}>

<label>

Enter a number:

<input

type="number"

value={number}

onChange={handleNumberChange}

/>

</label>

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

</form>

{isArmstrong !== null && (

<p>

{isArmstrong

? `${number} is an Armstrong number.`

: `${number} is not an Armstrong number.`}

</p>

)}

</div>

);

};

export default ArmstrongNumberChecker;

App.js

import React from "react";

import ArmstrongNumberChecker from "./ArmstrongNumberChecker";

const App = () => {

return (

<div>

<ArmstrongNumberChecker />

</div>

);

};

export default App;

**Q13. Write a react code to find out whether a number is a palindrome or not. The console must display "The number is a palindrome" if it is a palindrome. Otherwise it must display "The number is not a palindrome."**

**Sol.**

import React, { useState } from 'react';

function PalindromeChecker() {

const [inputValue, setInputValue] = useState('');

const [result, setResult] = useState('');

const checkPalindrome = () => {

const inputNumber = parseInt(inputValue);

if (isNaN(inputNumber)) {

setResult('Please enter a valid number');

return;

}

const reverseNumber = parseInt(inputValue.split('').reverse().join(''));

if (inputNumber === reverseNumber) {

setResult('The number is a palindrome');

} else {

setResult('The number is not a palindrome');

}

};

return (

<div>

<h2>Palindrome Checker</h2>

<input type="text" value={inputValue} onChange={(e) => setInputValue(e.target.value)} />

<button onClick={checkPalindrome}>Check</button>

<p>{result}</p>

</div>

);

}

export default PalindromeChecker;