**KONGU ENGINEERING COLLEGE, PERUNDURAI - 638 060**

**SEMESTER ODD|CONTINUOUS ASSESSMENT TEST – III**

**20ITE03 USER INERFACE DESIGN**

**ANSWER KEY**

**PART ­– A**

1. Explain the role of onClick event handler in react.

The onClick event handler is used to capture and respond to mouse clicks, typically on elements like buttons, links, or other interactive UI components

1. Compare ‘onClick’ and ‘onChange’ event handlers in react.

The onClick event handler is used to capture and respond to mouse clicks, typically on elements like buttons, links, or other interactive UI components. In contrast, the onChange event handler is used to capture and respond to changes in the value of input elements, such as text fields or checkboxes. It is commonly used for form handling and updating component state when the user interacts with input fields.

1. Define Uncontrolled Component.

Uncontrolled Components are the components that are not controlled by the React state and are handled by the DOM (Document Object Model). The useref hook is used to access any value that has been entered.

1. State the primary purpose of ‘useref’ hook in react.

To create mutable object references that persist across renders. Unlike state variables, changes to a ref object do not trigger a re-render of the component.

It can be used to access a DOM element directly.

1. Construct a React component that takes an array of course names as a prop and renders an unordered list (<ul>) with list items (<li>) for each course. Make sure to assign a unique key to each list item. Additionally, add a button that allows the user to add a new course to the list.

|  |
| --- |
| 1. Construct a React component that takes an array of course names as a prop and renders an unordered list (<ul>) with list items (<li>) for each course. Make sure to assign a unique key to each list item. Additionally, add a button that allows the user to add a new course to the list.   import React, { useState } from 'react';  const CourseList = ({ courses }) => {  const [newCourse, setNewCourse] = useState('');  const [courseList, setCourseList] = useState(courses);  const handleAddCourse = () => {  if (newCourse.trim() !== '') {  setCourseList([...courseList, newCourse]);  setNewCourse('');  }  };  return (  <div>  <ul>  {courseList.map((course, index) => (  <li key={index}>{course}</li>  ))}  </ul>  <div>  <input  type="text"  placeholder="Enter new course"  value={newCourse}  onChange={(e) => setNewCourse(e.target.value)}  />  <button onClick={handleAddCourse}>Add Course</button>  </div>  </div>  );  };  export default CourseList;  import CourseList from './Components/CourseList';  const App = () => {  const initialCourses = ['Math', 'Science', 'History'];    return (      <div>        <h1>Course List</h1>        <CourseList courses={initialCourses} />      </div>    );  };  export default App; |

1. Compare ‘useState’ hook and ‘useEffect’ hook in react.

useState Hook:

Used for adding state to functional components

Syntax:

Const [state, setState] = useState (initialState);

useEffect

Used to synchronize a component with an external system.

Syntax:

useEffect (setup, dependencies)

1. Identify an example of how to use the ‘useEffect’ hook to fetch and display a list of students from an API when the component mounts.

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

const StudentList = () => {

const [students, setStudents] = useState([]);

useEffect(() => {

const fetchStudents = async () => {

try {

const response = await fetch('https://api.example.com/students');

if (response.ok) {

const data = await response.json();

setStudents(data);

} else {

console.error('Failed to fetch data');

}

} catch (error) {

console.error('Error fetching data:', error);

}

};

// Call the fetchStudents function when the component mounts

fetchStudents();

}, []); // The empty dependency array ensures that the effect runs only once, similar to componentDidMount

return (

<div>

<h2>Student List</h2>

<ul>

{/\* Map through the list of students and display their names \*/}

{students.map(student => (

<li key={student.id}>{student.name}</li>

))}

</ul>

</div>

);

};

export default StudentList;

1. Tell the significance of the dependency array in the ‘useEffect’ hook.

The useEffect [manages an array that contains the state](https://codedamn.com/news/reactjs/useeffect-dependency) variables or functions which are kept an eye for any changes. These changes then trigger the callback function.

1. Recall the basic rule for using hooks in React and provide an example of a built-in hook that follows this rule.

Hooks should only be called at the top level of your React function components or custom hooks, not inside loops, conditions, or nested functions. This ensures that hooks are called in the same order on every render.

Hooks should only be called from within React function components or custom hooks. They should not be called from regular JavaScript functions or class components.

When using multiple hooks in a component, it should be called in the same order on every render.

1. What is Redux? Why it is used in react applications?

Redux is a state management library for JavaScript applications, and it is commonly used in React applications. The primary purpose of Redux is to manage the state of an application in a predictable and centralized way.

**PART B**

1. i) Summarize the key differences between controlled components and uncontrolled components in React.

| **Controlled  Component** | **Uncontrolled Component** |
| --- | --- |
| The component is under control of the component’s state. | Components are under the control of DOM. |
| These components are predictable as are controlled by the state of the component. | Are Uncontrolled because during the life cycle methods the data may loss |
| Internal state is not maintained | Internal state is maintained |
| It accepts the current value as props | We access the values using refs |
| Controlled by the parent component. | Controlled by the DOM itself. |
| Have better control on the form data and values | Has very limited control over form values and data |

11.ii) Explain the purpose of keys in React when rendering lists. Why are they important, and what problems do they help solve? Provide an example to illustrate their usage.

The purpose of keys is to help React identify which items have changed, been added, or been removed in a list. Keys are essential for efficient rendering and reconciliation of the virtual DOM.

**Importance:**

Efficient Updates

Preserving Components State

import React, { useState } from 'react';

const ItemList = () => {

const [items, setItems] = useState([

{ id: 1, text: 'Item 1' },

{ id: 2, text: 'Item 2' },

{ id: 3, text: 'Item 3' },

]);

const removeItem = (itemId) => {

setItems(items.filter(item => item.id !== itemId));

};

return (

<ul>

{items.map(item => (

<li key={item.id}>

{item.text}

<button onClick={() => removeItem(item.id)}>Remove</button>

</li>

))}

</ul>

);

};

export default ItemList;

1. Develop and implement controlled components for a user registration form for an online learning platform. The form collects essential information from users, such as their name, email, password, and a few preferences for customizing their learning experience. Create controlled components for each form field, include the use of state variables, event handlers, and any data validation.

import React, { useState } from 'react';

const RegistrationForm = () => {

// State variables for form fields

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

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

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

const [showPreferences, setShowPreferences] = useState(false);

const [preferredLanguage, setPreferredLanguage] = useState('English');

const [receiveNewsletter, setReceiveNewsletter] = useState(false);

// Event handler for form submission

const handleSubmit = (e) => {

e.preventDefault();

// Perform form submission logic (e.g., send data to server)

console.log('Form submitted:', { name, email, password, preferredLanguage, receiveNewsletter });

};

// Function for basic email validation

const validateEmail = (email) => {

// A simple email validation regex

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

return emailRegex.test(email);

};

// Function for basic password validation

const validatePassword = (password) => {

// Password should be at least 8 characters long

return password.length >= 8;

};

return (

<form onSubmit={handleSubmit}>

<div>

<label htmlFor="name">Name:</label>

<input

type="text"

id="name"

value={name}

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

required

/>

</div>

<div>

<label htmlFor="email">Email:</label>

<input

type="email"

id="email"

value={email}

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

required

onBlur={() => {

if (!validateEmail(email)) {

alert('Please enter a valid email address.');

}

}}

/>

</div>

<div>

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

<input

type="password"

id="password"

value={password}

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

required

onBlur={() => {

if (!validatePassword(password)) {

alert('Password should be at least 8 characters long.');

}

}}

/>

</div>

<div>

<label>

<input

type="checkbox"

checked={showPreferences}

onChange={() => setShowPreferences(!showPreferences)}

/>

Show Preferences

</label>

</div>

{showPreferences && (

<div>

<label htmlFor="preferredLanguage">Preferred Language:</label>

<select

id="preferredLanguage"

value={preferredLanguage}

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

>

<option value="English">English</option>

<option value="Spanish">Spanish</option>

<option value="French">French</option>

</select>

</div>

)}

<div>

<label>

<input

type="checkbox"

checked={receiveNewsletter}

onChange={() => setReceiveNewsletter(!receiveNewsletter)}

/>

Receive Newsletter

</label>

</div>

<div>

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

</div>

</form>

);

};

export default RegistrationForm;

1. Demonstrate a React component that includes a login form with email and password input fields. When the user submits the form, it should send a POST request to an authentication API with the provided credentials. If the authentication is successful, display a "Welcome, [user's name]" message; otherwise, show an error message.

import React, { useState } from 'react';

const LoginForm = () => {

// State variables for form fields and authentication status

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

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

const [isAuthenticated, setIsAuthenticated] = useState(false);

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

// Event handler for form submission

const handleSubmit = async (e) => {

e.preventDefault();

try {

// Send a POST request to the authentication API

const response = await fetch('https://api.example.com/authenticate', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ email, password }),

});

if (response.ok) {

// Authentication successful

const data = await response.json();

setIsAuthenticated(true);

setErrorMessage('');

alert(`Welcome, ${data.user.name}!`);

} else {

// Authentication failed

const errorData = await response.json();

setIsAuthenticated(false);

setErrorMessage(errorData.message);

}

} catch (error) {

console.error('Error during authentication:', error);

setIsAuthenticated(false);

setErrorMessage('An error occurred during authentication.');

}

};

return (

<div>

<h2>Login</h2>

<form onSubmit={handleSubmit}>

<div>

<label htmlFor="email">Email:</label>

<input

type="email"

id="email"

value={email}

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

required

/>

</div>

<div>

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

<input

type="password"

id="password"

value={password}

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

required

/>

</div>

<div>

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

</div>

</form>

{isAuthenticated && <p>Welcome, {email}!</p>}

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

</div>

);

};

export default LoginForm;

1. Construct a React component that utilizes the useEffect hook to set up an event listener for the mousemove event on the window object. When the mouse is moved, the component should update the state with the current mouse position (x and y coordinates) and display them on the screen in real-time. Ensure that you remove the event listener when the component unmounts.