Module – 9 React.js

1. Question : what is react.js? How is it different from other java script frameworks and libraries?

Answer:

* React is a java script library for building user interfaces (UI).
* It uses a component-based structure for reusable code.
* React updates the UI efficiently using a virtual DOM.
* JSX --- ( java script xml extensible markup language).
* JSX lets you write HTML like code in java script.
* It is faster compared to java script because it use hooks.

different from other java script frameworks and libraries.

* **React is a library,** not a full framework, meaning only handles the user interface, while other tools handle the rest.
* It uses **components** to break the UI into smaller, reusable pieces of code.
* React update the UL **efficiently** with a virtual DOM, making it faster than regular java script.
* It’s **flexible** and can be combined with other libraries or frameworks.

1. Question : Explain the core principles of React such as the virtual DOM and component-based architecture.

Answer:

**Virtual DOM:**

The virtual DOM is a lightweight, in-memory representation of the real DOM.

**Component-based :**

React builds UIs using components – independent, reusable pieces of code that return JSX (HTML + JS).

Components are like functions that return HTML elements.

1. Question : What are the advantages of using React.js in web development?

Answer:

1. Fast performance: React uses a virtual DOM , which makes update and rendering faster.
2. Reusable components: you can create components that can be reused, saving time and effort.
3. Esay to learn: React’ s syntax, especially with JSX, simple and intuitive for developers.
4. Large community: React has a big community, so you get lots of support and resources.
5. Scalability: React makes it easier to scale up your application as it grows.
6. Cross-platform:

JSX (JavaScript XML)

1. Question : What is JSX in React.js? Why is it used?

Answer:

* JSX means JavaScript XML (Extensible Markup Language)
* JSX stands for JavaScript XML.
* JSX allows us to Write HTML in React.
* JSX makes it easier to write and add HTML in React.
* JSX code is copied into JavaScript by React.

1. Question : How is JSX different from regular JavaScript? Can you write JavaScript inside JSX?

Answer:

* "JavaScript needs to be written inside the <script> tag in an HTML file, but JSX can't be written directly in HTML files. JSX is written in React and React converts it into JavaScript. JSX code is written in a JavaScript file."
* "Yes, you can write JavaScript inside JSX. To do this, you use curly braces {} around JavaScript expressions, like {name} or {2 + 2}, which inserts dynamic values into JSX."

1. Question : Discuss the importance of using curly braces {} in JSX expressions

Answer:

* In JSX, {} are used to include JavaScript expressions.
* They allow you to display variables, functions, or calculations.
* Example: <p>Hello, {name}! </p> will show dynamic content.
* Without {}, JSX treats content as plain text.
* Curly braces help mix logic and UI together.

Components (Functional & Class Components)

1. Question: What are components in React? Explain the difference between functional components and class components.

Answer 1:

* Components are independent and reusable bites of code.
* Components are reusable pieces of code that define a specific part of User Interface.

**Difference between Functional Components and Class Components:**

|  |  |
| --- | --- |
| **Function Components** | **Class Components** |
| Defined using a regular JavaScript Function. | Defined using ES6 Class. |
| Managed using the useState hook. | Managed using this.state. |
| Uses useEffect for lifecycle behavior. | Uses lifecycle methods like componentsDidMount. |
| Can use hooks like useState, useEffect. | Not applicable. |

2)

1. Question: How do you pass data to a component using props?

Answer 2:

* In the parent component, pass data as attributes to the child component:
  + <ChildComponent data="some data" />
* In the child component, access the data through props:
  + function ChildComponent(props) {
    - return <div>{props.data}</div>;

}

* The value of data is passed and used in the child component.

1. Question : What is the role of render() in class components?

Answer 3:

* The render() method in class components returns the JSX to be displayed on the screen.
* It's required in every class component and defines what the component should look like.

Props and State

1. Question : What are props in React.js? How are props different from state?

Answer 1:

* Props are inputs passed from a parent component to a child component.
* They are allowing it to receive data and behave dynamically.
* They are read-only and cannot be modified by the child component.

**Props Different from State:**

* Props come from the parent component while state is managed inside the component.
* Props are read-only, but state can change over time.
* Props are used for passing data While state is used for handling dynamic changes.

1. Question : Explain the concept of state in React and how it is used to manage component data.

Answer 2:

* State in React is an object that holds data specific to a component. It allows components to store and manage information that can change over time. When the state updates, react re-renders the component to reflect the changes.

1. Question : Why is this.setState() used in class components, and how does it work?

Answer 3:

* this.setState() is used in class components to update the state. When called, it changes the component’s state and triggers a re-render, so the component updates with the new state. It ensures the UI stays in sync with the state changes.
* this.setState() updates the current state. When you call setState(), React merges the new state with the old state and triggers a re-render of the component to reflect the changes.

Handling Events in React

1. Question: How are events handled in React compared to vanilla JavaScript? Explain the concept of synthetic events.

| **Aspect** | **Vanilla JavaScript** | **React** |
| --- | --- | --- |
| Syntax | addEventListener("click", ...) | <button onClick={handler}> |
| Event Type | Native DOM Events | SyntheticEvent (React wrapper) |
| Cross-browser Issues | Possible | Handled by React |
| Delegation | Manual | Automatic via root listener |
| Cleanup | Manual | Handled on component unmount |

Answer 1:

1. Question : What are some common event handlers in React.js? Provide examples of onClick, onChange, and onSubmit.

Answer 2:

* **Common Event Handlers in React.js:**
* React provides various event handlers to handle user interactions. Some of the most common ones includes:

1. onClick -Happens when a user clicls on something(like a button).
2. onChange – Happens when the value in an input box changes.
3. onSubmit – Happens when a user submits a form.

* **Examples:**
* onClick(Button Click)

A computer screen with text

AI-generated content may be incorrect.

* onChange(Input Change)

A screen shot of a computer program

AI-generated content may be incorrect.

* onSubmit(Form Submit)

A screen shot of a computer program

AI-generated content may be incorrect.

Question 3: Why do you need to bind event handlers in class components?

Answer 3:

You need to bind event handlers in class components to **ensure this refers to the class instance** (i.e., the component). Functional components using hooks (like useState, useEffect) avoid this issue entirely.

Conditional Rendering

1. Question : What is conditional rendering in React? How can you conditionally render elements in a React component?

Answer 1:

* **Conditional Rendering in React:**
* Conditinal rendering in React means showing or hiding elements based on certain conditions.

1. Explain how if-else, ternary operators, and && (logical AND) are used in JSXfor conditional rendering.

Answer 2:

| **Use Case** | **Syntax** | **When to Use** |
| --- | --- | --- |
| Multiple outcomes | if-else | Before return, for full control |
| Choose between two options | Ternary ? : | Inline JSX |
| Render if condition is true | Logical AND && | Show a piece of JSX conditionally |

Lists and Keys

1. Question : How do you render a list of items in React? Why is it important to use keys when rendering lists?

Answer 1:

**How to Render a List of Items in React**

In React, you render a list by using JavaScript’s .map() function inside JSX to transform an array of data into an array of React elements.

**Why Is It Important to Use Keys When Rendering Lists in React?**

In React, **keys** are used to uniquely identify elements in a list. They help React **track which items have changed, been added, or removed**, so it can update the UI efficiently during re-renders.

1. Question : What are keys in React, and what happens if you do not provide a unique key?

Answer 2

React uses a virtual DOM to compare changes between renders. Keys help React:

* Identify which list items have **changed**, been **added**, or **removed**.
* **Optimize re-rendering** by only updating affected elements.
* Maintain **correct component state** for elements like forms, inputs, or animations.

Forms in React

1. Question : How do you handle forms in React? Explain the concept of controlled components.

Answer 1:

* To handle forms in React, you typically use **controlled components** where the form elements' values are managed by React state.
* **Handling Forms in React:**

1️ **Use State to Control Input Values**

* Store form values in React state using useState.

2️ **Update State on Input Change**

* Use onChange to update state whenever the user types.

3️ **Prevent Default Form Submission**

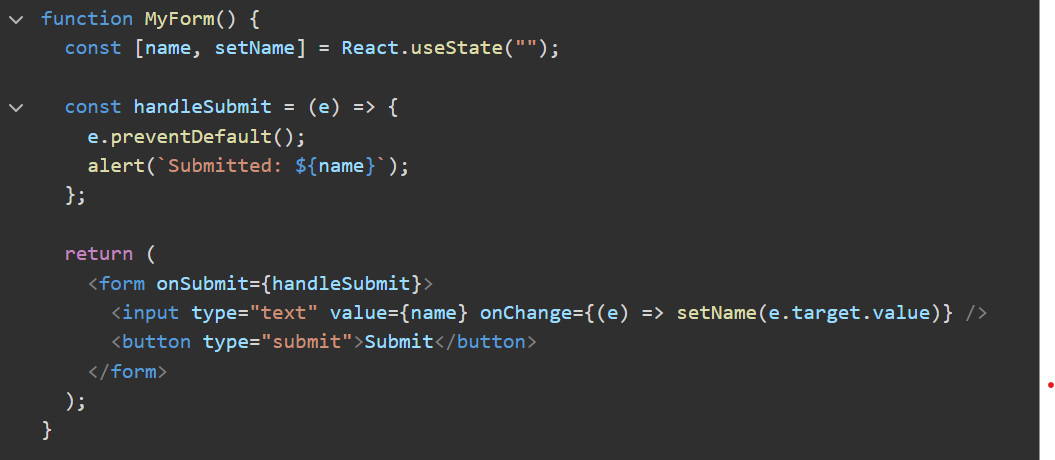
* Use e.preventDefault() to stop page reload on submit.

4️ **Submit Handler for Processing Data**

* Handle form submission inside an onSubmit function.
* **Controlled Components**:
* Definition:

A form element (input,textarea,select) whose value is controlled by React state.

* Why Use:
* Single Source of Truth – React manages imput values.
* Validation & Formatting – Modify values before submittimg.
* Better Control – Easily reset or update inputs dynamically.
* Example:



1. Question: What is the difference between controlled and uncontrolled components in React?

Answer 2:

**Difference between controlled and uncontrolled components:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Controlled Component** | **Uncontrolled Component** |
| Value Storage | Stored in React state | Stored in DOM (default behavior) |
| Updates | Controlled by usestate and onChange | Directly modified by the user |
| Access | State variable(value prop) | useRef (ref.current.value) |
| Best For | Form validation, real-time updates | Simple forms,uncontrolled inputs |
| Example Input | <input value={state} onChange={handler} /> | <input ref={inputRef /> |

* **Which One to Use?**
* Use Controlled Components for forms with validation,dynamic behaviour, or conditional rendering.
* Use Uncontrolled Components for simple forms where you only need the value on submit (e.f., file uploads).
* Controlled = React manages state
* Uncontrolled = Browser manages state

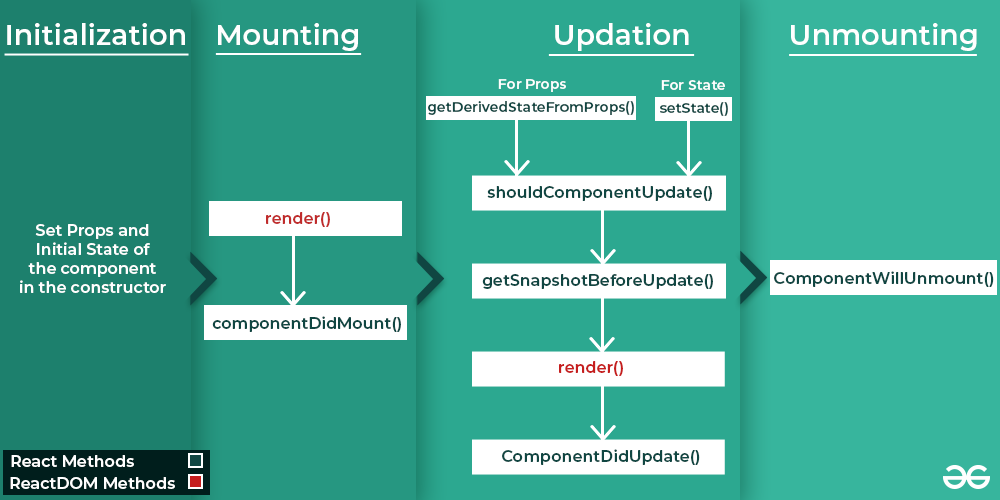
Lifecycle Methods (Class Components)

1. Question : What are lifecycle methods in React class components? Describe the phases of a component’s lifecycle.

Answer 1

* In react class components, **lifecycle methods** are special functions that let you run code at different stages of a component’s life. A component has three main phases in its lifecycle.

1. Mounting
2. Updating
3. Unmounting



* **Initialization: ( when the component is created )**
  + these methods are called the Set Props and initial State of the component in the constructor.
* Constructor
* **Mounting: ( when the component is added to the DOM )**
  + These methods are called in following order when an instance of components is being inserted into the DOM.
* **Render()**
* **componentsDidMount()**
* **updating: ( when the components state or props change, causing it to re-render)**
  + An update can be caused by changes to props or state. These methods are called in the following order when a component is being re-rendered:
* Static getDerivedStateFromProps()
* shouldComponentUpdate()
* getSnapshotBeforeUpdate()
* **render()**
* componentDIdUpdate()
* **unmounting: ( when the component is removed )**
  + this method is called when a component is being removed from the DOM.
* componentWillUnmount()

1. Question: Explain the purpose of componentDidMount(), componentDidUpdate(),and componentWillUnmount().

Answer 2

**Explanation of lifecycle methods:**

1. **componentDidMount()**

* **when it runs :** after the component is addedto the DOM (rendered on the screen).
* **Purpose:** to set up anything your component needs after its visible.
* **Uses :**
* Fetching data from an API.
* starting a timer.
* Subscribing to events.

1. **componenetDidUpdate()**

* **when it runs:** After the component updates (re-rendered) because of changes in its props or state.
* **Purpose:** to perform action based on updates.
* **Uses:**
* Making an API call when props/state change.
* Updating the DOM after changes.
* Reacting to user interaction.

1. **componentWillUnmount():**

* **when it run:** right before the component is removed from the DOM.
* **Purpose:** to clean up resources and prevent memory leaks.
* **Uses:**
* Clearing timers.
* unsubscribing from events.
* Canceling network requests.

These methods are key for managing side effects in class components !

Hooks (useState, useEffect, useReducer, useMemo, useRef, useCallback)

1. Question : What are React hooks? How do useState() and useEffect() hooks work in functional components?

Answer 1

* React hooks are tools that let functional components do more. They let you use features like **state ( data that changes)** and **lifecycle methods** ( action at different times ) without needing class components.
* **Hooks key overview:**
  + UseState: for state management.
  + useEffect: for handling side effect ( e.g., data fetching , subscription ).
  + Others: useContext, useReducer, useRef, etc…
* **useState()**
* adds and manage data ( state ) in component.
* The component when updates when the data changes.
* Example: track button clicks or toggle themese
* **useEffect()**
* handle tasks outside the component( e.g, fetching data or starting a timer).
* Runs on render or when specified values change.
* You can clean up tasks when the component is removed.
* **In shorts:**
* **useState():** manage data.
* **useEffect():** handle side tasks.

1. Question : What problems did hooks solve in React development? Why are hooks considered an important addition to React?

Answer 2:

### **Problems Hooks Solved in React Development:**

* **State management in functional components**: Before hooks, only class components could have state. Hooks like useState allow functional components to manage state.
* **Reusability of logic**: Hooks like useEffect let you reuse logic between components without changing their structure.
* **Cleaner code**: They remove the need for lifecycle methods in class components, simplifying code.
* **Easier to understand**: Hooks make functional components more powerful, reducing complexity compared to class components.

### **Why Hooks Are Important**:

* **Simplify Development**: Hooks make functional components as powerful as class components, but with less complexity.
* **Cleaner Code**: They reduce boilerplate code, making components easier to read and maintain.
* **Better Reusability**: Custom hooks allow logic to be reused across components.
* **Improved Side Effect Management**: useEffect() helps keep side effects clean and easy to manage in one place.

In short, hooks make React development simpler, more efficient, and easier to maintain!

1. Question : What is useReducer ? How we use in react app?

Answer 3

useReducer is a **React Hook** used to manage **complex state logic** in functional components. It's an alternative to useState, especially useful when:

* State depends on **previous state**.
* You have **multiple related state values**.
* You want **centralized state management** (similar to Redux in a simplified form).

1. Question: What is the purpose of useCallback & useMemo Hooks?

Answer 4:

**Purpose of useCallback & useMemo Hooks:**

* useCallback:
* useCallback memorizes a function, so it doesn’t get recreated on every render.
* Useful when passing function as props to child componens(prevents unnecessary re-renders).
* useMemo:
* useMemo memorizes a computed value, so React doesn’t recalculate it on every render.
* Useful for expensive calculations, filtering, or sorting large lists.

**Difference:**

|  |  |  |
| --- | --- | --- |
| **Hook** | **Optimizes** | **Use Case** |
| useCallback | Functions | Prevents function recreation |
| useMemo | Values/Calculations | Avoids unnecessary recalculations |

1. Question 5: What’s the Difference between the useCallback & useMemo Hooks?  
   Answer 5:

**Difference between the useCallback & useMemo:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **useCallback** | **useMemo** |
| What it does? | Memoizes a function | Memoizes a computed value |
| Returns | A function | A value(computed result) |
| Use case | Prevents unnecessary function recreation | Avoids expensive recalculations |
| Common usage | Passing stable functions to child components | Optimizing heavy calculations |

**Simple Explanation:**

* useCallback:
* Used when you want to memorize a function and avoid re-creating it on every render.
* useMemo:
* Used when you want to memorize a value so that React doesn’t recompute it unnecessarily.

1. Question : What is useRef ? How to work in react app?

Answer 6:

* It can be used to store a mutable value that does not cause a re-render when updated.
* It can be used to access a DOM element directly.

**Work in react app:-**

1️ **Accessing DOM Elements** – Like focusing an input box directly without using state.  
2️ **Storing Previous Value** – Like tracking the previous value of a counter without re-rendering.  
3️ **Does Not Trigger Re-render** – Unlike useState, updating a useRef value does not cause the component to re-render.

**Or**

1️ **DOM Element ko Access Karna** – Jaise ek input box ko directly focus karna, bina state use kiye.  
2️ **Previous Value Store Karna** – Jaise ek counter ki purani value track karna, bina re-render kiye.  
3️ **Re-render Trigger Nahi Karta** – Agar aap useState use karein toh har update pe component re-render hota hai, par useRef me aisa nahi hota.