React Notes

1. **What is React?**

React is a front-end and open source JavaScript Library, which used to build complex and reusable user interface components of mobile and web application as it follows component based approach.

1. **Important Features:**

* It supports server side rendering
* It will make use of Virtual DOM rather than real DOM
* It follows unidirectional data binding
* It uses reusable components for developing the view.

1. **What is React Hooks?**

React hooks are functions, that enable functional component to use React State and lifecycle features.

1. **Hook Rules:**

* Hooks can only be called inside React function components.
* Hooks can only be called at top level of a component.
* Hooks cannot be conditional.

1. **What are Keys in React?**

* Keys in React are special attribute used to identify the items in a list.
* Keys are unique strings that help to identify which elements are added, updated and remove from react list.
* Without keys, react does not understand the order or uniqueness of each element.

1. **What is JSX**

JSX stands for JavaScript XML.

It allows us to write HTML inside JavaScript and place them in DOM without using functions like appendChild () or createElement ().

1. **What is Virtual DOM**

The Virtual DOM is a programming concept, where virtual representation of a UI is kept in a memory and synced with the real DOM by library such as ReactDOM.

Virtual DOM is a virtual representation of Real DOM.

1. **How Does the Virtual DOM Work?**

The Virtual DOM works by creating an in-memory representation of the Real DOM, allowing developers to make changes without directly affecting the actual page. Here’s how it works:

* Virtual DOM Creation: When the application state changes, the Virtual DOM creates a new virtual tree.
* Diffing Algorithm: The new Virtual DOM tree is compared with the previous version, and the differences (known as “diffs”) are calculated.
* Batch Updates: Instead of updating the entire Real DOM, only the nodes that have changed are updated, leading to faster rendering.

1. **Difference between functional & class components**

|  |  |
| --- | --- |
| Functional Component | Class Component |
| * A functional components is just a plain JS pure function, that accepts props as argument and returns a React element (JSX) | * A class component requires to extend from React. Component and create a render function that returns a React element. |
| * Functional components are declared using arrow function or function keyword | * Class component are declared using ES6 class. |
| * There is no render() used in functional component | * It must have the render () method for returning JSX |
| * Known as Stateless components, as they simply accepts data and display it | * Known as Stateful components, as they implement logic and state. |
| * React lifecycle methods cannot be used   Ex. componentDidMount | * React lifecycle methods can be used inside class component.   Ex. componentDidMount |

1. **What are props in React?**

Props are arguments passed into React components

React props are like function arguments in JavaScript and attributes in HTML

|  |  |
| --- | --- |
| Props | State |
| * Data is passed from one component to another | * Data is passed within component only |
| * It is Immutable (cannot be modified) | * It is mutable (can be modified) |
| * Props are read-only | * The state is both read and write |
| * Props can be access by child component | * State cannot be access by child component. |

1. **What is Props drilling?**

Prop drilling is occurring when, a props needs to be passed through several layers of nested components to reach deeply nested child component that actually needs props.

Each intermediary component in a hierarchy has to pass the prop down, even if it doesn’t use the prop itself.

1. **Context API:**

* To address the issue of props drilling, context api is used.
* It allows to share the information with different components that are not directly connected.
* contextAPI consist of two properties that is provider and consumer. The provider provides a state to the children, and the consumer uses the state has been pass.
* **How to Use:**
  1. **Create context** – use createContext to define a context
  2. **Wrap Component tree** – wrap the component tree with Context Provider.
  3. **Access Context Value** – In child component, useContext to access context value

1. **React Hooks:**

React Hooks are built in functions, that permit developers for using the state and lifecycle methods within React component.

Using Hook, all feature of React can be used without writing class component.

1. **Why were Hooks are introduced in React?**

React hooks were introduced in the 16.8 version of React.

Previously, functional components were called stateless component. Only class component was used for state management and lifecycle method.

The need to change a functional component to a class component, whenever state management or lifecycle methods were to be used, led to the development of Hooks.

1. **What are the rules that must be followed while using React Hooks?**

* Hooks can only be called inside React function components.
* Hooks can only be called at the top level of a component.
* Hooks cannot be conditional

1. **useState Hook**

useState hook is a special function that lets you to add state to functional components.

It provides a way to declare and manage state variable directly within functional component.

1. **useEffect Hook**

The useEffect in react is used to handle the side effects such as fetching data and updating DOM.

This hook runs on every render but there is also a way of using dependency array using which we can control the effects of rendering.

1. **How to use componentWillUnmount & componentMount with Functional Components in React**

import React, {useEffect} from 'react';

const ComponentExample => () => {

    useEffect (() => {

        // Anything in here is fired on component mount.

        return () => {

  // Anything in here is fired on component unmount.

        }

    }, [])

}

1. **different types of dependency array effects in react**
2. **No Dependency Array:**

Executes the effect **after every render** of the component.

1. **Empty Dependency Array**

Executes the effect only **once** after the initial render.

1. **Single Dependency**

Executes the effect whenever the specified dependency changes.

1. **Multiple Dependencies**

Executes the effect whenever **any dependency** in the array changes

1. **useMemo Hook**

It is used to get the memorized value of a function in react components.

The useMemo Hook returns a memorized value and prevents the application from unnecessary re-renders.

Syntax:

const memoizedValue = useMemo (functionThatReturnsValue, arrayDependencies)

1. **Difference between useEffect, useMemo, useCallback hooks?**

* **useEffect** is used for handling side effects and executing code after rendering or when certain dependencies changes
* **useMemo** is used for memoize the result of a function or expression to prevent unnecessary recalculations.
* **useCallback** is used to memoize functions, especially useful for optimizing child components that depend on callback functions.
* *All three hooks take an array of dependencies. When these dependencies change, useEffect runs its effect function, useMemo recomputes the value, and useCallback recreates the memoized function.*

1. **useRef Hook**

useRef is a Hook provided by React, and it is commonly used for accessing and interacting with DOM elements. It can also be used for preserving values between renders without causing re-renders.

1. **useReducer Hook**

The useReducer Hook is similar to the useState Hook.

A useReducer is a hook in React that allows you add a reducer to your component. It takes in the reducer function and an initialState as arguments. The useReducer also returns an array of the current state and a dispatch function.

**Syntax:** const [state, dispatch] = useReducer (reducer, initialState);

* **state**: represents the current value and is set to the initialState value during the initial render.
* **dispatch**: is a function that updates the state value and always triggers a re-render, just like the updater function in useState.
* **reducer**: is a function that houses all the logic of how the state gets updated. It takes state and action as arguments and returns the next state.
* **initialState**: houses the initial value and can be of any type.

1. **Explain the use of render method in React?**

* React renders HTML to the web page by using function called render ().
* The purpose of the function is to display the specified HTML code inside the specified HTML element.
* In the render method, we can read props and state and return our JSX code to the root component of our app.
* In the render method, we cannot change the state, and we cannot cause side effects (such as HTTP request to the webserver).

1. **What is higher-order component in React?**

HOC is the advanced method of reusing the component functionality logic.

It simply takes the original component and returns the enhance component.

HOC are beneficial as they are easy to code and read. Also, helps to get rid of copying the same logic in every component.

1. **What is spread operator in React?**

The Spread operator (…) in react, is used to manage and manipulate the props, state and arrays.

What is map()

Create a simple React component that fetches data from an API and displays it, handling loading and error states.