Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

Conditional Rendering in React

Conditional rendering in React allows you to render different UI elements or components based on certain conditions. It's similar to conditional statements in JavaScript, such as if, else, and ternary operators. React uses these conditional statements to decide which elements to render.

Key Methods for Conditional Rendering

1. **if Statement:**
    - Use an if statement to conditionally render components.

```
let showComponent = false;
if (showComponent) {
  return <ClassBased />;
} else {
  return <FunctionBaased />;
}
```

Ternary Operator:

- A more concise way to conditionally render elements.

```
 let showComponent = false;
return showComponent ? <ClassBased /> : <FunctionBaased />
```

React Hooks

Introduction

React Hooks are functions that allow you to "hook into" React features from functional components. They were introduced in React 16.8 to provide state and lifecycle features in functional components, which were previously only possible in class components. Hooks offer a more direct API to the React concepts you already know and love.

Purpose of React Hooks

1. **State Management**:
    - Allows functional components to manage state using useState.
2. **Side Effects**:
    - Manage side effects (such as data fetching and subscriptions) using useEffect.
3. **Context**:
    - Access and use context values with useContext.
4. **Ref Management**:
    - Interact with DOM elements or persist values across renders with useRef.
5. **Complex State Logic**:
    - Handle complex state logic with useReducer.

6. **Custom Hooks**:
    - Encapsulate and reuse stateful logic by creating custom hook

useState in React

useState is a hook in React that allows you to add state to functional components. Introduced in React 16.8, hooks provide a way to use state and other React features without writing class components.

Key Features of useState

1. **State Management in Functional Components**:
    - Before hooks, state management was only possible in class components. useState allows functional components to manage their own state.
2. **Initialization**:
    - The useState hook takes the initial state as an argument and returns an array with two elements: the current state and a function to update it.
3. **Re-rendering**:
    - When the state is updated using the function returned by useState, the component re-renders to reflect the new state.

```
const [state, setState] = useState(initialState);
```

- state: The current state value.
- setState: A function to update the state.
- initialState: The initial value of the state.

Example

Here's a simple example of how to use the useState hook in a functional component:

```
import { useState } from "react";
import "../../Component/ClassBased.css";
import { useState } from "react";

let StateManage = () => {
  let [state, setState] = useState(0);
  let handleChange = () => {
    setState(prevState => prevState + 1);
  }
return (
  <div className="train-info">
    <div>{state}</div>
    <button onClick={handleChange}>Count Status</button>
  </div>
);
};
export default StateManage;
```

Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

Interview Questions:

What is conditional rendering in React?

- **Answer:** Conditional rendering in React allows you to render different UI elements or components based on certain conditions, similar to how conditional statements work in JavaScript.

How can you use the ternary operator for conditional rendering?

- **Answer:** The ternary operator provides a concise way to conditionally render elements. Example:

```
let showComponent = false;
return showComponent ? <ClassBased /> : <FunctionBaased />
```

What are React Hooks?

- **Answer:** React Hooks are functions that let you use state and other React features in functional components. They allow you to "hook into" React's state and lifecycle features from function components without needing to write class components.

Why were Hooks introduced in React?

- **Answer:** Hooks were introduced to solve several problems:
  - Eliminate the need for class components to manage state and lifecycle.
  - Promote code reuse through custom hooks.
  - Simplify state and side effect management in functional components.
  - Improve code readability and maintainability.

What is the useState hook in React?

- **Answer:** The useState hook is a function that allows you to add state to functional components. It returns an array with two elements: the current state value and a function to update that state.

How do you initialize state using the useState hook?

- **Answer:** You initialize state by calling useState with an initial value.

```
let [state, setState] = useState(0);
```

How can you update the state using the useState hook?

- **Answer:** You update the state by calling the function returned by useState

```
  setState(state + 1);
```

What is the purpose of the functional update form in useState?

- **Answer:** The functional update form allows you to update the state based on the previous state value, ensuring you have the most recent state. E

```
setState(prevState => prevState + 1);
```

Can you explain the difference between setting state directly and using the functional update form?

- **Answer:** Setting state directly `setState(state + 1);`may lead to stale state issues when multiple updates are batched. The functional update form `setState(prevState => prevState + 1);` ensures the update is based on the latest state.