1. What is Redux?

**Redux** is a **predictable state container** for JavaScript applications.

**State Management Library.**

**Global State.**

**Compatible with Various Libraries and Frameworks.**

1. What is Redux Thunk used for?

**Redux Thunk** is a middleware for **Redux** that allows you to return **functions**, rather than just actions, within Redux. Let’s dive into its purpose and use cases:

**Handling Asynchronous Actions.**

**Delayed Logic Execution.**

**Writing Thunks.**

**Common Use Cases.**

To set up Redux Thunk, install it using npm install redux-thunk --save or yarn add redux-thunk**.**

1. What is Pure Component? When to use Pure Component over Component?

PureComponent is a class in React that extends the base Component class. It provides some key optimizations related to re-rendering behavior.

When you create a component as a PureComponent, it automatically handles the shouldComponentUpdate lifecycle method for you.

     import React from 'react';

class MyPureComponent extends React.PureComponent {

render() {

return <h1>Welcome to GeeksforGeeks</h1>;

}

}

// Regular Component (for comparison)

class MyRegularComponent extends React.Component {

render() {

return <h1>Welcome to GeeksforGeeks</h1>;

}

}

1. What is the second argument that can optionally be passed tosetState and what is its purpose?

The second argument that can optionally be passed to setState in React is a **callback function**. Its purpose is to execute code immediately after the state update is completed and the component has been re-rendered.

Here’s why this callback function is useful.

**Asynchronous Nature of setState.**

**Using the Callback Function.**

**Example-**

        import React, { Component } from 'react';

class MyComponent extends Component {

  constructor(props) {

    super(props);

    this.state = {

      name: 'GFG',

    };

  }

  updateName = (value) => {

    console.log('Previous name:', this.state.name);

    this.setState({ name: value }, () => {

      console.log('Current name:', this.state.name);

    });

  };

  render() {

    const { name } = this.state;

    return (

      <div>

        <p>Welcome To GFG</p>

        <input

          type="text"

          value={name}

          onChange={(e) => this.updateName(e.target.value)}

        />

      </div>

    );

  }

}

export default MyComponent;