Q1. What are components in React? Explain the difference between functional components and class components

Ans: Components are the basic building blocks of a React application. A component represents a reusable piece of the user interface, such as a button, form, header, or footer. Each component contains its own logic and UI, which helps in building applications that are modular, reusable, and easy to maintain.

In React, components are mainly of two types: Functional Components and Class Components.

1. Functional Components

Functional components are simple JavaScript functions that return JSX. They are easier to write and understand and are widely used in modern React applications.

Features of Functional Components:

- Written as JavaScript functions
- Use **Hooks** (like useState, useEffect) to manage state and lifecycle
- Less code and better readability
- Faster and simpler than class components

Example:

```
function Welcome () {
  return <h1>Hello, World </h1>;
}
```

2. Class Components

Class components are ES6 classes that extend React.Component. They were traditionally used to manage state and lifecycle methods before hooks were introduced.

Features of Class Components:

- Written using ES6 class syntax
- Use this.state to manage data
- Use lifecycle methods like componentDidMount()
- More complex and longer syntax

Example:

```
class Welcome extends React.Component {
   render () {
      return <h1>Hello, World</h1>;
   }
}
```

Difference Between Functional and Class Components

Feature	**Functional Components**	**Class Components**
Syntax	JavaScript function	ES6 class
State Management	Using Hooks	Using this.state
Lifecycle Methods	Using Hooks	Built-in lifecycle methods
Code Length	Short and simple	Longer and complex
Performance	Better	Slightly slower
Usage	Preferred in modern React	Used in older React code

Q2. How do you pass data to a component using props?

Ans: In React, props (properties) are used to pass data from a parent component to a child component. Props allow components to be dynamic, reusable, and configurable. They are read-only, meaning a child component cannot modify the props it receives.

Steps to Pass Data Using Props

1. Pass data from the parent component

Data is passed to a child component as attributes when the component is used.

Example (Parent Component):

```
function App() {
   return <Welcome name="Nagesh" age={23} />;
}
```

2. Receive data in the child component

The child component receives the data through the props object.

Example (Child Component):

```
function Welcome(props) {
   return (
      <h1>
         Hello, {props.name}! You are {props.age} years old.
      </h1>
   );
}
```

Using Destructuring with Props

Props can also be destructured for cleaner and more readable code.

```
function Welcome ({ name, age }) {
   return <h1>Hello, {name} You are {age} years old. </h1>;
}
```

Q3. What is the role of render () in class components?

Ans: In React class components, the render() method is a mandatory method that defines what should be displayed on the screen. It returns JSX, which represents the UI of the component.

Whenever a component's state or props change, React automatically calls the render() method again to update the user interface.

Key Roles of render() Method

1. Returns JSX (UI Structure)
   The render() method returns JSX that describes how the component should look.

Example:

```
class Welcome extends React.Component {
   render() {
      return <h1>Hello, World</h1>;
   }
}
```

2. Controls UI Updates
   When state (this.state) or props (this.props) change, React re-runs the render() method to reflect the updated data on the screen.

3. Does Not Modify State
   The render() method should be pure, meaning it should not change state or perform side effects. It only reads data and returns JSX.

4. Handles Conditional Rendering
   Logic such as conditions and expressions can be used inside the render() method to control what is displayed.