##### ****1. Explain React Components****

React components are the building blocks of a React application's UI. Each component is a piece of the user interface, and multiple components can be composed together to build complex UIs.

Components allow developers to break down the UI into independent, reusable parts and think about each part in isolation.

### ****2. Differences Between Components and JavaScript Functions****

|  |  |  |
| --- | --- | --- |
| Feature | JavaScript Function | React Component |
| Purpose | Performs logic or computation | Renders UI |
| Return Type | Returns values (numbers, strings) | Returns JSX (HTML-like elements) |
| Integration with React | Not inherently part of React | Core part of React |
| Lifecycle Methods | Not available | Available (in class components) |
| State Management | No built-in state | Can use useState or this.state |

### ****3. Types of Components****

React has two main types of components:

1. **Class Components**
2. **Function Components**

### ****4. Class Component****

A class component is a traditional way to define a component using ES6 classes. It can hold and manage state and use lifecycle methods.

**Example:**

import React, { Component } from 'react';

class Welcome extends Component {

render() {

return <h1>Hello from a Class Component</h1>;

}

}

export default Welcome;

**More Examples:**

// Home.js

import React, { Component } from 'react';

class Home extends Component {

render() {

return (

<div>

<h1>Welcome to the Home Page of Student Management Portal</h1>

</div>

);

}

}

export default Home;

// About.js

import React, { Component } from 'react';

class About extends Component {

render() {

return (

<div>

<h1>Welcome to the About Page of Student Management Portal</h1>

</div>

);

}

}

export default About;

// Contact.js

import React, { Component } from 'react';

class Contact extends Component {

render() {

return (

<div>

<h1>Welcome to the Contact Page of Student Management Portal</h1>

</div>

);

}

}

export default Contact;

### ****5. Function Component****

Function components are simpler and are written using plain JavaScript functions. They can use hooks like useState and useEffect.

**Example:**

function Welcome() {

return <h1>Hello from a Function Component</h1>;

}

export default Welcome;

### ****6. Component Constructor****

The **constructor** is a special method used in class components to initialize state and bind event handlers.

**Syntax:**

constructor(props) {

super(props);

this.state = {

name: "React"

};

}

* super(props) calls the parent class constructor.
* State is initialized using this.state.

### ****7. render() Function****

The **render()** method is mandatory in a class component. It describes what to display on the screen (usually returns JSX).

**Example:**

render() {

return (

<div>

<h1>This is rendered from render()</h1>

</div>

);

}

* Every class component must define a render() method.
* It must return a single React element (usually JSX).

### ****8. Component Lifecycle in React****

#### ****Need and Benefits of Lifecycle Methods****

* Lifecycle methods help you run code at particular times in a component’s life.
* Useful for tasks like data fetching, manual DOM manipulation, subscriptions, timers, and cleanup.
* They ensure better performance and resource management.

### ****9. Lifecycle Hook Methods****

|  |  |  |
| --- | --- | --- |
| Phase | Method | Purpose |
| Mounting | constructor() | Initialize state |
|  | static getDerivedStateFromProps() | Sync state with props |
|  | render() | Render UI |
|  | componentDidMount() | Run after component is mounted |
| Updating | static getDerivedStateFromProps() | Sync state with props again |
|  | shouldComponentUpdate() | Decide whether to re-render |
|  | render() | Re-render component |
|  | getSnapshotBeforeUpdate() | Capture some info before update |
|  | componentDidUpdate() | Run after component updates |
| Unmounting | componentWillUnmount() | Cleanup before component is removed |

### ****10. Sequence of Steps in Rendering a Component****

1. **constructor()** is called to initialize the state.
2. **getDerivedStateFromProps()** is called to update state based on props.
3. **render()** is called to render JSX.
4. **componentDidMount()** runs after the component is added to the DOM.
5. On state or prop changes:
   * **getDerivedStateFromProps()** again
   * **shouldComponentUpdate()**
   * **render()**
   * **getSnapshotBeforeUpdate()**
   * **componentDidUpdate()**
6. When component is removed:
   * **componentWillUnmount()**