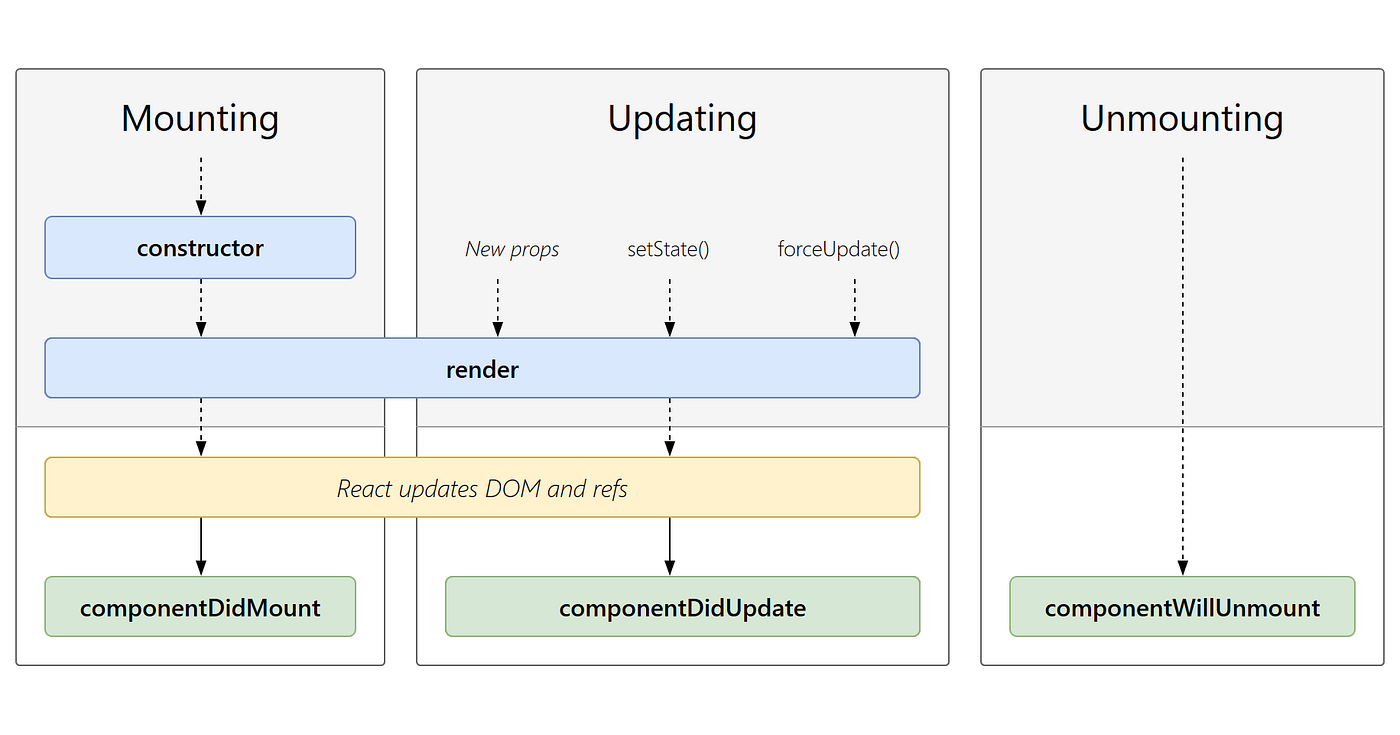
**MODULE: 4 (List and Hooks)**

**Q : Explain Life cycle in Class Component and functional component with Hooks**

**Ans :**

Brief about life cycle in class component and functional component is shown below.



**Class component and functional component includes:**

**Mounting:**

The mounting phase refers to the initial setup and rendering of a component. During the mounting phase, specific lifecycle methods are called for both class component and functional components with hooks.

ComponentWillMount():- This method is called before the component mounts on DOM or we can say before render method is called.

ComponentDidMount():- this method is called after the component mounted on DOM or we can say after render method is called.

**Updating:**

The updating phase refers to the period during which a component is re –rendered due to changes in its state or props. This phase occurs after the initial rendering and involve updating the component to reflect the new state or prop values.

ComponentShouldUpdate():- This method is called when you want to re-render the component on some condition . By default, it returns true.

ComponentWillUpdate():-This method called before re-rendering has done and when you want to do some functionalities just before re-rendering take place.

ComponentDidUpdate():- This method is called after the re-rendering has done or we can say after component mounted on DOM.

**Unmounting:**

The unmounting phase in React occurs when a component is being removed from the DOM. During this phase, cleanup tasks and actions can be performed before the component is completely taken out of the application.

ComponentWillUnmount():-This Method is called before the component Unmounting on DOM. Denoting that it will Unmounted soon or we can say end of the component life cycle.

**Example:-**

1. **import** React, { Component } from 'react';
3. **class** App **extends** React.Component {
4. constructor(props) {
5. **super**(props);
6. **this**.state = {hello: "React.js"};
7. **this**.changeState = **this**.changeState.bind(**this**)
8. }
9. render() {
10. **return** (
11. <div>
12. <h1>ReactJS component's Lifecycle</h1>
13. <h3>Hello {**this**.state.hello}</h3>
14. <button onClick = {**this**.changeState}>Click Here!</button>
15. </div>
16. );
17. }
18. componentWillMount() {
19. console.log('Component Will MOUNT!')
20. }
21. componentDidMount() {
22. console.log('Component Did MOUNT!')
23. }
24. changeState(){
25. **this**.setState({hello:"All!!- Its a great reactjs tutorial."});
26. }
27. componentWillReceiveProps(newProps) {
28. console.log('Component Will Recieve Props!')
29. }
30. shouldComponentUpdate(newProps, newState) {
31. **return** **true**;
32. }
33. componentWillUpdate(nextProps, nextState) {
34. console.log('Component Will UPDATE!');
35. }
36. componentDidUpdate(prevProps, prevState) {
37. console.log('Component Did UPDATE!')
38. }
39. componentWillUnmount() {
40. console.log('Component Will UNMOUNT!')
41. }
42. }
43. export **default** App;