**React Assignment**

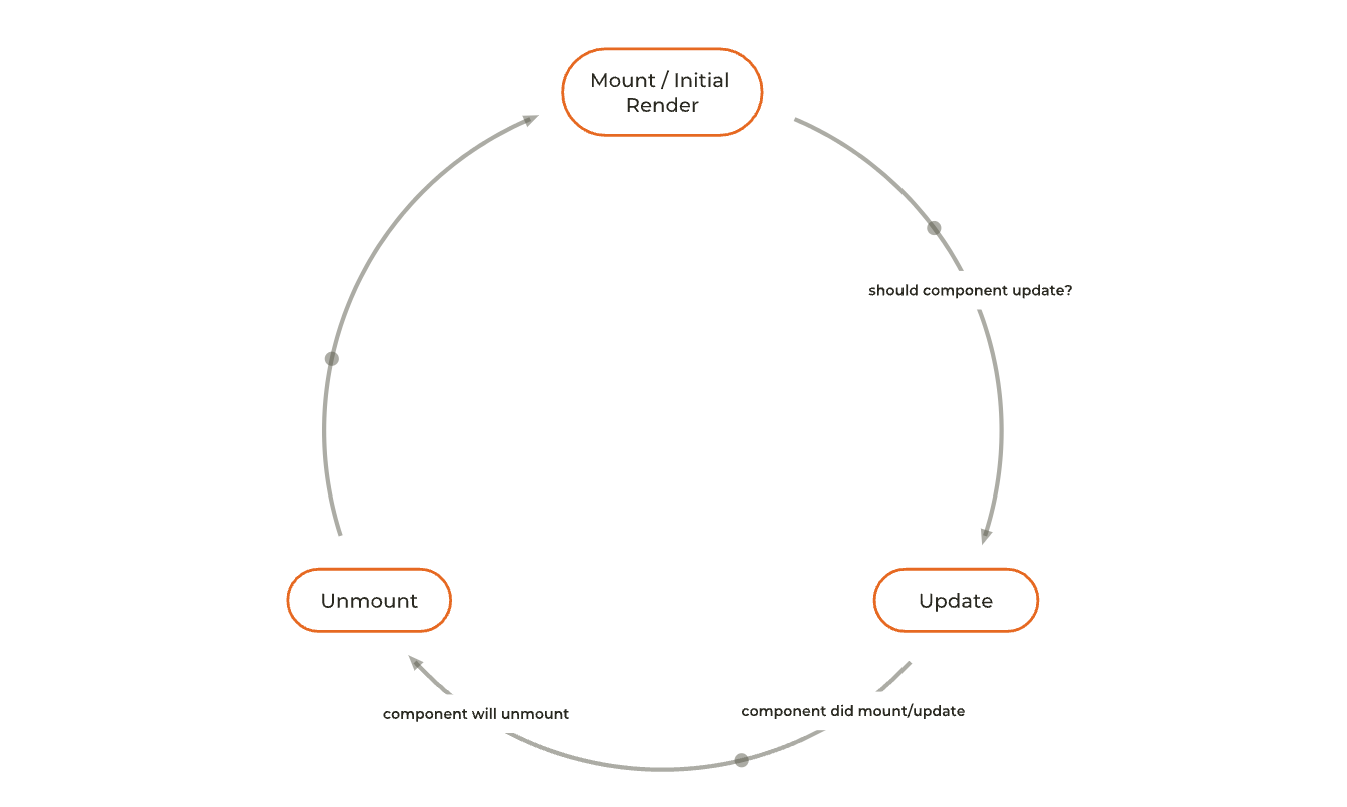
**Module-4 (List and Hooks)**

**Q1:** **Explain Life cycle in Class Component and functional component with Hooks**

**Ans: There are three Lifecycle methods in ReactJs.**

Life cycle of react component

1. **Mount** (Initial render)
2. **Update** (When the state Used in the component or props to the added to the component is changed)
3. **Unmount**



A React component undergoes three phases in its lifecycle: mounting, updating, and unmounting.

1. The mounting phase is when a new component is created and inserted into the DOM or, in other words, when the life of a component begins. This can only happen once, and is often called “initial render.”
2. The updating phase is when the component updates or re-renders. This reaction is triggered when the props are updated or when the state is updated. This phase can occur multiple times, which is kind of the point of React.
3. The last phase within a component's lifecycle is the unmounting phase, when the component is removed from the DOM.

In a class-based component, you can call different methods for each phase of the lifecycle (more on this below). These lifecycle methods are of course not applicable to functional components because they can only be written/contained within a class. However, react hooks give functional components the ability to use states.

Hooks have gaining popularity because they make working with React cleaner and often less verbose.

import React from "react";

import ReactDOM from "react-dom/client";

class Test extends React.Component {

    constructor(props) {

**super**(props);

**this**.state = { hello: "World!" };

    }

    componentDidMount() {

        console.log("componentDidMount()");

    }

    changeState() {

**this**.setState({ hello: "React" });

    }

    render() {

**return** (

            <div>

                <h1>

                    Hello

                    {**this**.state.hello}

                </h1>

                <h2>

                    <a

                        onClick={**this**.changeState.bind(

**this**

                        )}

                    >

                        Press Here!

                    </a>

                </h2>

            </div>

        );

    }

    shouldComponentUpdate(nextProps, nextState) {

        console.log("shouldComponentUpdate()");

**return** **true**;

    }

    componentDidUpdate() {

        console.log("componentDidUpdate()");

    }

}

const root = ReactDOM.createRoot(

    document.getElementById("root")

);

root.render(<Test />);

**Example of class component and functional component in react life cycle.**

**Functional Components:**

Functional components are some of the more common components that will come across while working in React. These are simply JavaScript functions. We can create a functional component to React by writing a JavaScript function.

**Example of functional component**

import React, { useState } from "react";

const FunctionalComponent = () => {

    const [count, setCount] = useState(0);

    const increase = () => {

        setCount(count + 1);

    }

**return** (

        <div style={{ margin: '50px' }}>

            <h3>Counter App using Functional Component : </h3>

            <h2>{count}</h2>

            <button onClick={increase}>Add</button>

        </div>

    )

}

export **default** FunctionalComponent;

**Class Component**

This is the bread and butter of most modern web apps built in ReactJS. These components are simple classes (made up of multiple functions that add functionality to the application).

**Example of class component**

import React, { Component } from "react";

class ClassComponent extends React.Component {

    constructor() {

**super**();

**this**.state = {

            count: 0

        };

**this**.increase = **this**.increase.bind(**this**);

    }

    increase() {

**this**.setState({ count: **this**.state.count + 1 });

    }

    render() {

**return** (

            <div style={{ margin: '50px' }}>

                <h3>Counter App using Class Component : </h3>

                <h2> {**this**.state.count}</h2>

                <button onClick={**this**.increase}> Add</button>

            </div>

        )

    }

}

export **default** ClassComponent;