LifeCycle In ReactJS

1) Lifecycle In Class Components

In React has a lifecycle which We can monitor and manipulate during its Four phases.

The Four phases are: **Mounting**, **Updating**, Error Handling And **Unmounting**.

Mounting Phase

Mounting means putting elements into the DOM. React has four built-in methods that gets called, in this order, when mounting a component:

1. constructor()
2. getDerivedStateFromProps()------Rare Use In React
3. render()
4. componentDidMount()

1) Constructor() Method

-The constructor() method is called before anything else, when the component is initiated, and it is the natural place to set up the initial state and other initial values.

-The constructor() method is called with the props, as arguments, and you should always start by calling the super(props) before anything else, this will initiate the parent's constructor method and allows the component to inherit methods from its parent (React.Component).

2) getDerivedStateFromProps() Method

-The getDerivedStateFromProps() method is called right before rendering the element(s) in the DOM.

-This is the natural place to set the state object based on the initial props.

-It takes state as an argument, and returns an object with changes to the state.

3) render() Method

-The render() method is required, and is the method that actually outputs the HTML to the DOM.

4) componentDidMount() Method

-The componentDidMount() method is called after the component is rendered.

-This is where you run statements that requires that the component is already placed in the DOM.

LifeCycle In ReactJS

Updating Phase

The next phase in the lifecycle is when a component is *updated*.

A component is updated whenever there is a change in the component's state or props.

React has five built-in methods that gets called, in this order, when a component is updated:

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

Note:- The render() method is required and will always be called, the others are optional and will be called if you define them.

1) ComponentDidUpdate

-The componentDidUpdate method is called after the component is updated in the DOM.

- this function gets invoked once after the render() function is executed after the updation of State or Props.

UnMounting Phase

-The next phase in the lifecycle is when a component is removed from the DOM, or *unmounting* as React likes to call it.

-React has only one built-in method that gets called when a component is unmounted:

1) componentWillUnmount()

-The componentWillUnmount method is called when the component is about to be removed from the DOM.

LifeCycle In ReactJS

Hooks

-The two most common default hooks are useState and useEffect. The useState hook gives state to the functional component, and useEffect allows you to add side effects within it (like after initial render), which aren't allowed within the function's main body. You can also act upon updates on the state with useEffect.

1) useState()

-The useState hook is used to store state for a functional component. This hook accepts one parameter: initialState, which will be set as the initial stateful value, and returns two values: the stateful value, and the update function to update the stateful value. The update function accepts one argument, newState, which replaces the existing stateful value.

2)useEffect()

-The **useEffect** hook works similarly to the three lifecycle methods: **componentDidMount**, **componentDidUpdate**, and **componentWillUnmount**.

-As with the render() method of a class-based component, the main body of a functional component should be kept pure. With the useEffect hook, you're able to create side effects while maintaining the component's purity. Within this hook, you can send network requests, make subscriptions, and change the state value.

The Syntax Of UseEffect:-

```
useEffect(()=>{
        Block Of Code
},[])
```