**Lifecycle of React Native Application Component**

Introduction

Hello Guys, Today we will talk about the Lifecycle of React Native Application Component. Lifecycle methods are inbuilt methods. As a Mobile Application developer, we have to take care of the Lifecycle of each screen/activity/component because sometimes we have to load the data accordingly. For example, if I want to initialize some connection when the user opens the screen reader and closes it when the user closes it; In both cases we have to have some knowledge about the Lifecycle methods so that we can perform such actions.

The lifecycle of React Native Application

There are 4 types of Lifecycle methods available in React Native:

1. Mounting methods
   1. constructor()
   2. componentWillMount() (Deprecated after RN 0.60)
   3. render()
   4. componentDidMount()
2. Updating methods
   1. componentWillReceiveProps() (Deprecated after RN 0.60)
   2. shouldComponentUpdate()
   3. componentWillUpdate() (Deprecated after RN 0.60)
   4. componentDidUpdate()
3. Unmounting methods
   1. componentWillUnmount()
4. Error handling methods
   1. componentDidCatch()

**Description**

**1. Mounting Methods**

**1. constructor():** It is the first method called when we open a screen, it is mostly used to create States.

**constructor**() {

**super**();

console.log('Constructor Called.');

}

**2. componentWillMount():** It is called right after constructor(), used to call asynchronous tasks or network calls.

componentWillMount() {

console.log('componentWillMount called.');

}

**3. render():** Render is the most important Lifecycle method because it is used to render UI or we can say the main View of the screen.

render() {

**return** (

<**View** style={{ justifyContent: 'center', alignItems: 'center' }}>

<**Text**>Language is: {this.props.name}</**Text**>

</**View**>

);

}

**4. componentDidMount():** Is called after render method, It is used for the network calls.

componentDidMount() {

console.log('componentDidMount called.');

}

**2. Updating methods**

Updating methods are used to update the value of Props or State to React Native. These methods are called automatically when a component re-renders.  
**1. componentWillReceiveProps():** It is called before the component dose anything with new props, We would send the next prop as an argument inside it.

componentWillReceiveProps(nextProp) {

console.log('componentWillReceiveProps called.', nextProp);

}

**2. shouldComponentUpdate():** It is called every time before the screen or parent component re-renders. We can stop the re-rendering of the screen by passing false in this method.

shouldComponentUpdate(nextProp, nextState) {

console.log('shouldComponentUpdate called.');

**return** true;

}

**3. componentWillUpdate():** It is called before the re-rendering when new state or props is received for updating. It does not allow the this.setState({}) method.

componentWillUpdate(nextProp, nextState) {

console.log('componentWillUpdate called.');

}

**4. componentDidUpdate():** It is called after the rendering, this method is mostly used to interact with updated rendering values and execute any post-render events.

componentDidUpdate(prevProp, prevState) {

console.log('componentDidUpdate called.');

}

**3. Unmounting method**

**1. componentWillUnmount():** It is called when the component is removed from the DOM, Users can clear any running timers, stop network requests and cleaning any previously stored value in the application in this method.

componentWillUnmount() {

console.log('componentWillUnmount called.');

}

**4. Error handling method**

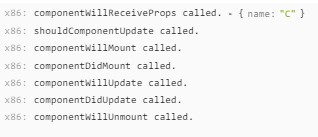
**1. The componentDidCatch():** It is a part of the error handling method. It is used to find errors between JavaScript code and handle them by passing the correct message or argument. It will help us to use any try /cache block to handle any error.

componentDidCatch(error, info) {

console.log('componentDidCatch called.');

}

That was the different methods that are called in a Lifecycle of any component. Below is the example which will help you to understand more

/\*Example of Reac Native Life Cycle\*/

import React, { Component } from 'react';

import { Text, View } from 'react-native';

class CustomComponent extends Component {

constructor() {

super();

console.log('Constructor Called.');

}

componentWillMount() {

console.log('componentWillMount called.');

}

componentDidMount() {

console.log('componentDidMount called.');

}

componentWillReceiveProps(nextProp) {

console.log('componentWillReceiveProps called.', nextProp);

}

shouldComponentUpdate(nextProp, nextState) {

console.log('shouldComponentUpdate called.');

return true;

}

componentWillUpdate(nextProp, nextState) {

console.log('componentWillUpdate called.');

}

componentDidUpdate(prevProp, prevState) {

console.log('componentDidUpdate called.');

}

componentWillUnmount() {

console.log('componentWillUnmount called.');

}

componentDidCatch(error, info) {

console.log('componentDidCatch called.');

}

render() {

return (

<View style={{ justifyContent: 'center', alignItems: 'center' }}>

<Text>Language is: {this.props.name}</Text>

</View>

);

}

}

export default class App extends Component {

render() {

return (

<View style={{ flex: 1, justifyContent: 'center' }}>

<CustomComponent name="C" />

</View>

);

}

}