Application LifeCycle Methods

React Native comes with React and react has its own Application Life Cycle Methods which is used in React Native. The life cycle methods is the inbuilt function created by react native, These function will execute according to user requirement and execute at a particular steps of application.

There are basically 4 types of methods available in react native:

1. Mounting method.
2. Updating method.
3. Unmounting method.
4. Error handling method.

1. Mounting methods:

There are 4 types of mounting methods available in react native, i have explained each below.

1. constructor() : Constructor is always called at the first time when react native application will start in mobile, it is mostly used to create States in react native application class.

```
constructor(){
   super();
   console.log("Constructor Called.");
 }
```

2. componentWillMount() : This function is just called right after constructor() called, It is mostly used to call asynchronous functions or web calls from react native apps.

```
componentWillMount(){

   console.log("ComponentWillMount() Called.");
 }
```

3. render() : Render function is one of the most important function of a class because it is used to render View's or any graphical representation on screen using its return block. Without the use of render function in our class we cannot make stable Views or Screens in react native applications.

```
render()
 {
  console.log("Render Called");
  return(
   <View style = { styles.MainContainer }>
    <Text style={{fontSize: 20}}>React.Component</Text>
   </View>
  );
 }
```

4.componentDidMount() : componentDidMount function called itself after render called, It is used to call Web Calls to parse JSON data first time when application will start.

```
componentDidMount(){

  console.log("ComponentDidMount() Called.");

 }
```

2. Updating methods:

Updating methods is used to update or change the value of Props or State in react native. These components is automatically called when a pre built component is re-rendered.

1. componentWillReceiveProps() : This function would called before our component dose anything with new props, We would send the next prop as argument inside it.

```
componentWillReceiveProps(nextProps) {

  this.setState({

    value: nextProps.myProp + "hello"
```

```
  });
 },
```

2. shouldComponentUpdate() : The shouldComponentUpdate() function calls every time before the screen or parent component re-rendering process. You can stop re-rendering screen by passing false in this function.

```
shouldComponentUpdate(nextProp, nextState) {

  console.log(nextProp, nextState);

  console.log(this.props, this.state);

  return false;

 }
```

3. componentWillUpdate() : This function is however called before re-rendering process and when new state or props is received for updating and dose not allow the this.setState({}) method.

```
componentWillUpdate(nextProp, nextState) {
  console.log('componentWillUpdate Called', nextProp, nextState);
}
```

4. componentDidUpdate() : The componentDidUpdate() function called after the React updates the DOM, this method is mostly used to interact with updated DOM value and execute any post render events. You can use it with Library which directly interact with the DOM.

This method has its own 2 arguments:

- prevProps: previous properties object.
- prevState: previous state object.

3. Unmounting method:

This method is called just after when a component or View is removed from DOM. There is only 1 function in unmounting section.

1. componentWillUnmount() : This function is called right after the component is removed form DOM or destroyed, Users can clear any running timers, stop network requests and cleaning any previously stored value in application.

```
componentWillUnmount() {
      this.value= this.value.destroy();
  }
```

4. Error handling method :

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

```
componentDidCatch(error, info){

//Handle error.

}
```

React	React Native
	<View>
	<Text>
,	<ListView>
<img>	<Image>