**Lifecycle of Components:**

**Mount**: Specific methods which are provided to Dom, where Dom will render them and give this information to root.

Mount has 4 built in methods, which get called.

1. Constructor- When the component is initiated this method is called at first

import React, { Component } from 'react'

class lifecycle1 extends Component {

constructor(props) {

super(props)

this.state = {name: "Constructor"};

}

render() {

return (

<div>

<div> <h1> Life cycle of components {this.state.name}</h1> </div>

</div>

)

}

}

export default lifecycle1

1. **getDriveStateFromProps**: It is called after constructor and before rendering.

import React, { Component } from 'react'

class lifecycle1 extends Component {

constructor(props) {

super(props)

this.state = {name: "Constructor"};

}

static getDerivedStateFromProps(props, state) {

return {name: props.comp };

}

render() {

return (

<div>

<div> <h1> Life cycle of components {this.state.name}</h1> </div>

</div>

)

}

}

export default lifecycle1

1. **Render**- What we see on the UI, it is because of render function. It gives html to Dom

import React, { Component } from 'react'

class lifecycle1 extends Component {

render() {

return (

<div>

<div> <h1> Life cycle of components</h1> </div>

</div>

)

}

}

export default lifecycle1

1. **ComponentDidMount**- Statement that require component to be present in DOM can be written through this
2. class lifecycle1 extends Component {
3. constructor(props) {
4. super(props)
6. this.state = {name: "Constructor"};
7. }
8. componentDidMount() {
9. setTimeout(() => {
10. this.setState({name: "render"})
11. }, 1000)
12. }

15. render() {
16. return (
17. <div>
19. <div> <h1> Life cycle of components {this.state.name}</h1> </div>
21. </div>
23. )
24. }
25. }
26. export default lifecycle1

**Updating:** This will come into picture when components are updated. Basically, when there is a change in static and props components are updated.

To update component react uses 5 built-in method

1. getDrivedStatefromProps: This is state first method to be called to update compnent

class lifecycle2 extends React.Component {

constructor(props) {

super(props);

this.state = {subject: "science"};

}

static getDerivedStateFromProps(props, state) {

return {subject: props.train };

}

changeSub = () => {

this.setState({subject: "maths"});

}

render() {

return (

<div>

<h1>My Favorite subject is {this.state.subject}</h1>

<button type="button" onClick={this.changeSub}>Suject change</button>

</div>

);

}

}

1. shouldComponentUpdate: This return a Boolean value that specify react to render or not.

class lifecycle2 extends React.Component {

constructor(props) {

super(props);

this.state = {subject: "science"};

}

static getDerivedStateFromProps(props, state) {

return {subject: props.train };

}

shouldComponentUpdate() {

return false;

}

changeSub = () => {

this.setState({subject: "maths"});

}

render() {

return (

<div>

<h1>My Favorite subject is {this.state.subject}</h1>

<button type="button" onClick={this.changeSub}>Suject change</button>

</div>

);

}

}

1. render- After updating, new html has to be given to Dom i.e. re render.

class lifecycle2 extends React.Component {

constructor(props) {

super(props);

this.state = {subject: "science"};

}

}

render() {

return (

<div>

<h1>My Favorite subject is {this.state.subject}</h1>

<button type="button" onClick={this.changeSub}>Suject change</button>

</div>

);

}

}

1. getSnapshotBeforeUpdate:

component was mounted with a cricket, as the timer changes the state the sports changes to football.

class lifecycle2 extends React.Component {

constructor(props) {

super(props);

this.state = {sports: "cricket"};

}

componentDidMount() {

setTimeout(() => {

this.setState({sports: "football"})

}, 1000)

}

getSnapshotBeforeUpdate(prevProps, prevState) {

document.getElementById("div1").innerHTML =

"Before the update, the favorite was " + prevState.sports;

}

componentDidUpdate() {

document.getElementById("div2").innerHTML =

"The updated favorite is " + this.state.sports;

}

render() {

return (

<div>

<h1>My Favorite sports is {this.state.sports}</h1>

<div id="div1"></div>

<div id="div2"></div>

</div>

);

}

}

Unmounting: To unmount a component react has only one built-in function i.e. componentWillUnmount this will delete the component.

class lifecycle3 extends React.Component {

constructor(props) {

super(props);

this.state = {show: true};

}

delHeader = () => {

this.setState({show: false});

}

render() {

let myheader;

if (this.state.show) {

myheader = <Child />;

};

return (

<div>

{myheader}

<button type="button" onClick={this.delHeader}>Delete Header</button>

</div>

);

}

}

class Child extends React.Component {

componentWillUnmount() {

alert("The component named Header is about to be unmounted.");

}

render() {

return (

<h1>Hello World!</h1>

);

}

}