HANDWRITTEN NOTES

A Deep-Dive into ⟶

REACT-COMPONENT LIFECYCLE

Read it Now

OR

Save it for Later.

Author : Ashraya KK

Guide : Akshay Saini's

Namaste React Live Session

React component lifecycle refers to the series of methods that get executed at different stages of a component's existence in React application.

The lifecycle methods can be divided into 3 phases :→

1. Mounting Phase
2. Updating Phase
3. Unmounting Phase

Mounting Phase

→ These methods are called when an instance of a component is being created & inserted into the DOM :

- constructor()
- static getDerivedStateFromProps()
- render()
- componentDidMount() → This method is called immediately after the first render of of a component. It is executed only once during the lifecycle of a component.

Updating Phase :

→ These methods are called when a component is updated in response to changes in its props or state :-

- **shouldComponentUpdate()** -

  This method is called before a component is updated. It returns a boolean value indicating whether the component should be updated or not.

- **componentWillUpdate()** -

  This method is called just before a component is updated. It is only executed if shouldComponentUpdate() is true

- **render()** - Used to rended the component after it has been updated.

- **componentDidUpdate()** - This method is called immedietly after a component is updated. It is only executed if ShouldComponentUpdate() is true.

Unmounting Phase

—▷ The phase where component is being removed from the DOM.

- componentWillUnmount() :- This method is called just before a component is removed from the DOM.

Best Place to make API call in class Components

—▷ componentDidMount () { . . }

This is, because during mounting phase.

1) Constructor ( ) —▷ is called

then, 2) render ( ) —▷ is called.

atlast, 3) componentDidMount() → is called

Eg:-
```
class ApidatafetchExample extends React.Compment{
    state = { data: [], loading: true, error: null};
  componentDidMount (){
      this.fetchData (); }

  FetchData = ()=>{
    fetch ("https://api.example/data")
```

```
.then(response => response.json())
.then(data => this.setState({data, loading:false}));
. catch(data
. then(
. catch(error => this.setState({error,
                                loading: false}));
};
```

CORE - BASIC OF CLASS COMPONENT

About.js (Parent Component)

```
class About extends Component{
 constructor(props){
   super(props);
console.log("Parent-constructor");
 }
componentDidMount(){
console.log("Parent-componentDidMount");
 }
render(){
 console.log("Parent-render");
 return(
 <Profile />
 );}}
```

Profile.js (Child component)

```
class Profile extends Component{
 constructor(props){
   super(props);
console.log("Child-constructor");
 }
componentDidMount(){
console.log("Child-componentDidMount");
 }
render(){
 console.log("Child-render");
 return(
    <h1>Profile class</h1>
 );
}}
```

—▷ In above eg.

<About /> is the Parent component

<Profile /> is the child component of <About />

—▷ In what order the above code will execute?

1. Parent - Constructor
2. Parent - render
3. Child - constructor
4. Child - render
5. Child - componentDidMount
6. Parent - componentDidMount

(In render() of About it sees <Profile /> and it will trigger the lifecycle method of this childeren component)

Another Case

If <About/> component have 2 children :->

First child & Second child.

Let's see how it will be executed :-

About.js

```
class About extends Component {
    constructor (props){
        Super (props);
console.log("Parent-constructor");
}
componentDidMount () {
    console.log ("Parent- render componentDidMount");
}
render () {
    console.log ("Parent-render");
return ( <>
<Profile name = {"First child"} />
<Profile name = {"Second child"} />
</> ); }}
```

Profile class .js

```
class Profile extends React.component {
    constructor (props) {
        Super (props);
        this.state = {
            count : 0,
        };
    console.log ("Child - constructor " + this.props.name);
}
componentDidMount () {
    console.log ("Child - componentDidMount" +
                this.props.name);
}
render () {
    console.log ("Child - render" + this.props.name);
    return (
        <h1> Profile Class </h1>
    )
};
```

Order of Execution :-

1. Parent - Constructor
2. Parent - Render
3. First child - constructor
4. First child - Render
5. Second child - constructor
6. Second child - Render

COMMIT PHASE STARTS

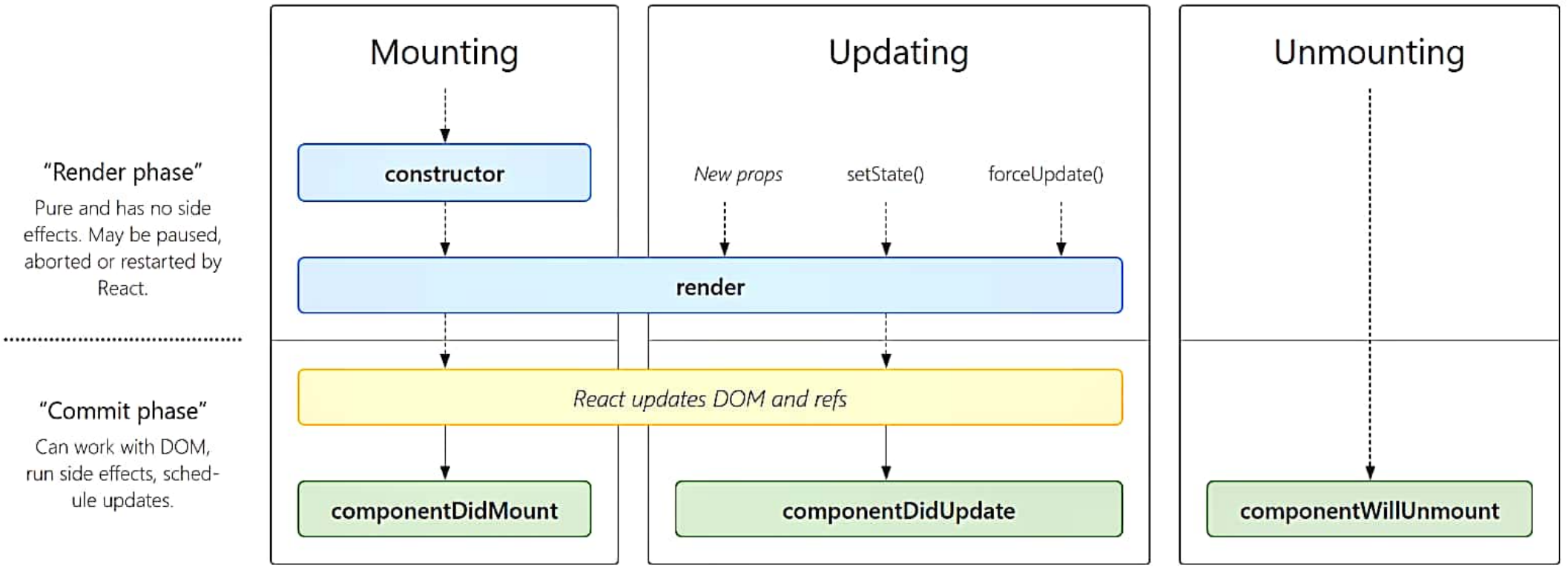
7. First child - componentDidMount
8. Second child - componentDidMount
9. Parent - componentDidmount

Explanation

See React lifecycle diagram ⟹

React completes the render phase of every child and after that move to commit phase.

Suppose after first-child-render, if I call first child-componentDidMount, it will make an api call and my ~~fir~~ second child will stay there. So, react will batch the render phase of first & second child



React do rendering in 2 phases :-

1. Render Phase
2. Commit Phase

First of all, react finishes the RENDER PHASE.

Render Phase : → is fast

→ includes constructor and render method.

Commit Phase

→ Phase where react modifies the DOM.

→ componentDidMount is called after the initial render has finished.

→ Commit Phase is slow.

Making an API Call

→ Let's use github user api.

→ make an api call in the child component

Profileclass.js :-

* class Profile extends React.component {

```
constructor(props) {
    Super (props);
        this.state = {
            userInfo : {
                name : " ",
                location : " ",
            } };
    Console.log ("Child - Constructor");
}
async componentDidMount () {
    const data = await fetch ("https://api-github..
                              .com /users /Ashrayaa");
    const json = await data.json ();
    console.log (json);
    this.setstate ({
        userInfo : json,
    });
    console.log ("Child - componentDidMount");
}
```

```
render() {
  console.log("Child - render");
  return (
    <h1> Name : {this.state.userInfo.name} </h1>
    <img src={this.state.userInfo?.avatar_url} </h1>
    <h2> Location : {this.state.userInfo.location}
                                              </h2>
```

Sequence of method called on above code

I have parent 'About.js' inside one child 'Profileclass.js'.

1. Parent - Constructor
2. Parent - Render
3. Child - constructor
4. Child - render
5. ~~API call~~
5. Parent - ComponentDidMount } is called before making api call.

This is because, React finishes render cycle first and then it goes to commit cycle. As Child - componentDidmount, will take some time for the data to load, Parent - componentDidMount is called before. So, hence this Sequence.

1. Parent - constructor
2. Parent render
3. Child constructor
4. child render
5. DOM is updated
6. json is logged in console
7. Parent - componentDidMount
8. Child - componentDidMount
   It is called before but is been put into the wait cycle. Because we are using async.
9. Child - render

* setState trigger next render. It will trigger reconciliation process. So, the child will be rendered once again when we have the data.

This re-render cycle is known as "UPDATING"