Ep-8 > Let's get classy

Class-based Component - it's the older way of Creating Component (Normal js class)

Functional based Component · Newer way of writing a Component

---

→ import React from "react";

```
keyword  Name of
         component
class UserClass extends React.Component {
      render () { }
}
```

render () → Render method

{ } → returns a piece of jsx to display on Webpage

⇒ Creating a Class based Component

```
export default Userclass;
```

```
Constructor (props){
   super(props);
}
```

super(props); → Remember to use this

⟹ Use Props in your class Component while Receiving props from parent

---

```
Const About = () =>{
   return (
      <>
        <User name={"xyz1"}/>
        <ClassUser name={"xyz2"}/>
      </>
   ),
}
```

Class based Component

```
import React from "react";

class ClassUser extends React.Component {
   constructor (props){
       super (props)
   }
   render() {
      return (
        <>
          <div> Name: {this.props.name}</div>
        </>
      )
   }
export default ClassUser;
```

Functional Component

```
Const User = ( {name} ) =>{
   return (
       <>
         <h2> Name : {name} </h2>
       </>
   )
};
export default User;
```

creating state variable in class Component

whenever a Instance of a class component is Created, a state Instance is Created.

↳ a Constructor is Created.

↳ Constructor is the best place to ~~create a class based Component~~

```
this.state = {
    count: 0,   --> state variable.   (All the state variable comes under object)
}
```

//NEVER EVER UPDATE STATE VARIABLE DIRECTLY

∴ HOW TO UPDATE STATE VARIABLE IN CLASS BASED COMPONENT.

```
this.setState = {
    count : this.state.count + 1,     (A big object is passed to setState and it will update the variables)
}
```

React Life Cycle

- When the class loads ⇒ New Instance of class is Created
  ↳ And when class is Instantiated
  ↳ Constructor is Called (1st)
- Once the Constructor is Called
  ⟹ then render() is called. (2nd)

Just like Constructor & render() method, class-based Component have one more method ⇒ ComponentDidMount(){} (3rd) ⇒ when the Component is loaded, First Constructor is called, then render() method is Called and Once the class based Component is mounted on to the DOM, then ComponentDidMount() is Called.

Parent-child life Cycle

1) Parent Construtor
2) Parent Render
3) Child Construtor
4) Child Render
5) child Component Did mount
6) Parent Component Did Mount

Both Are Same

import React from "react";
class Userclass extends React.Component {
}
	import {Component} from "react";
	class Userclass extents Component {
	}

Very Important Use Case of React life Cycle / class based Component

- componentDidMount is used make API call

Why we make API call inside ComponentDidMount ?

⇒ Because we want to Render the Component, then Make an API call and then fill the data. We want to make Render my Component as fast as possible, then make an API call & get the data.

⇒ In Class Based Component,

first $\underbrace{\text{Constructor will be Called, then Render}}$ will be Called

~~This Both will Render the Component~~

&

then ComponentDidMount() will be Called which is make to use API called

⟶ Hence first the Component is Rendered & then API is Called.

React life Cycle diagram

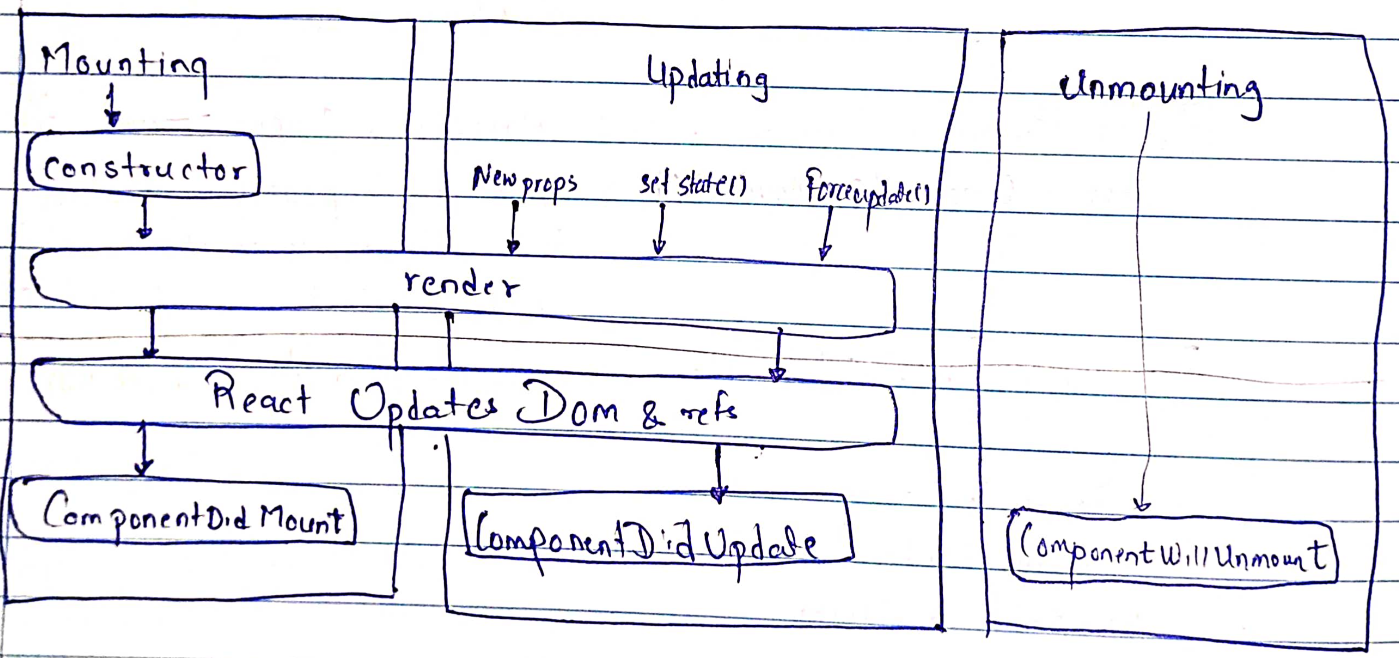
React works in 2 phases (Render phase) (Commit phase)

"Render phase"

ure & has no ide effects. ay be paused borted or restarted

Commit phase"

an work with Dom run side effects Schedule updates



Render phase is very fast

Commit phase takes time

because first step in Commit phase is Dom Manipulation

& Dom updating is expensive & it takes time.

- Virtual Dom is updated in SINGLE BATCH

- Nav Item Component

ComponentDidUpdate => Used we want to update the State variable or Display the Fetch API Data.
- It is also used to update the Component.

Component Will Unmount => Used we move from one Page to another page
- Component will get unmount when navigate to another page

Mounting
- Constructor (with dummy data)
- Render (with dummy data)
  <HTML dummy data)
- Component Did Mount
  - API call
  - <this.setstate -> State updated>

Update
- render (with API data)
- HTML <new API data>)
- ComponentDid Update

Unmount
- ComponentwillUnmount
(As soon as you navigate to other page)

NEVER EVER COMPARE (REACT LIFECYCLE) WITH (REACT FUNCTIONAL COMPONENT)