L-8

CLASSY

No longer use
Highly asked in Interviews.

Read about other Routers from react-router-dom official documentation.

Nested Routing :-
To configure path using react-router-dom we create relative path using children.

when assign path so, for children no need to use slash

Children: [{
path: "profile",
element }]

here no need to use slash

otherwise react considers it

we want localhost:1234/about/profile

localhost:1234/profile

CSS - Prism

Ctrl & Click

Better Comments

Prettier

Indent Rainbow

Auto Import

REST Client

Git Log

SAGAR

Page No.

Date

Slash means from the root

if write for children without slash
it works path: /about /
~~but without path~~ Slash

Children always render in outlet
outlet always create in parent

→ Class based Component

To At end it is normal
JS Class

→ We have to instruct react that it is
react class component not normal JS class
So, for that we use -

Ex class Profile extends React.Components {

}

This Component Come from React
by import React from 'react';

SAGAR
Page No.
Date

We cannot create class component without using Rendered method
// → render() { return // code }

This render method return JSX
↳ it is mandatory to use.

we have to also export class component
for ex:- export default Profile;

Some like functional component we have export & import in desired file where we want to render it.

extends :→ use to inherit some properties

→ In class base :- To this.Props. anything / help to get props

ex- render() {
return (
{this.Props.name}
)
}

What happen when React See Props in Class based Component

- So, React keep track of class, so it collect all Props of that class and attached to that class Now we can use it by using this

One thing it is not mandatory to use setamount in use state in functional component

Const [count] = useState('0') - as we are not using a setcount so, need to write

<h>{[count]}</h>

In class based component, it has constructor() {

H/w why we have two props
constructor(props) {
super(props);

For state variable

In class we use constructor(props) {
super(props);

}

Constructor is place that uses for initialization

Whenever class is render, constructor is call so, ~~state is bes~~ this is best place to create state.

we use this. to get props & now for state we use like this

```
constructor (props) {
  super (props);
  // create state
  this.state = {
    count=0;      default value
  };
```

to use this

<h1> {this.state.count} </h1>

In class based component all state variable created as part of one object.

So, for using more than one state

```
this.state = {
    count: 0,
    count2: 0,
}
```

React uses one big object to maintain whole state even in functional component also.

how do we setState (as like we have done in functional component) in class comp.

React gives us setState

```
<button onClick={() => {
    this.setState({
        count: 1,
    });
}}>
    setCount
</button>
```

we do not mutate state directly.

we never do this.state = something.

Page No.
Date

Suppose we have to update second count also
then simply

```
<button onclick={()=>{
        this.setState({
          count:1,
          count2:2,
        })
}}>
```

this also can be work but not best

```
<button onclick={()=>{
        this.setState(()=>{
          count:1,
        })
        this.setState(()=>{
          count:2;
        })
}}>
```

~~Note~~ from here 1st Read Assignment then Read this

In life cycle of Class

first of all Constructor call
then render

we know useEffect best for API to call
because whenever we can render
first then later on we update the
state by (API call).

→ In class there is a method to call render after render i.e
componentDidMount()

1st of all Constructor call
2nd render call
then componentDidMount call

so, the best place to call API is componentDidMount because it after render

This is life cycle method & there are lots of other life cycle method

V.V.I

We can also write
class React extends Component {
render() {
}
}

but for this we have to import
import {Component} from "react";

Fig

SAGAR
Page No
Date

→ React rendering in two phase
1st - Render
2nd - Commit phase

- React first Complete Render phase
Render phase include Render & Constructor

Commit phase where React modify dom

~~how React use bind~~

s/

Parent Constructor
Parent render
First Child Constructor
First child render
Second child constructor
Second child render
First Chid ComponentDom
Second child componentDomat
Post ChildComponentDomat

React first call Constructor then render and render its read to put things to dom so, In Commit Recat will update dom & then it will call ComponentDidMount

Page No.
Date

⇒ Render phase is fast than commit phase because in commit phase it has to update dom which is harder, render phase is dealing with object & creating html which is fast. React batches up inside render phase.

what happen is

First React →

Parent constructor
Parent render
first child constructor
first child render

batch of render done

React will say there are more children so let me first complete render phase then commit phase. as because suppose if one of child make api call then it will delay render of other child.

⇒ we can do async with componentDidMount but not with useEffect

⇒ componentDidUpdate is call after every update. cycle effect: → it is like useEffect

earlier

componentDidUpdate(prevProps, prevState) {

if (this.state.count !== prevState.count) {

// code

}

Take one:-

componentDidMount() {

this.timer = setInterval(() => {

console.log("Namaste")

}, 1000);

}

this will trigger when we visit page.

variable should be accessible in that class reference.

componentWillUnmount() {

clearInterval() {

clearInterval(this.timer);

console.log

}

This will trigger when we leave page.

1:51

This behaviour also shown in functional based compnet so,

```
useEffect(() => {
  const timer = setInterval(() => {
    console.log('Hi')
  }, 1000);

  // to cancel it
  return () => {
    clearInterval(timer)
  }
```

it is cleanup function which excecut when leave base the ComponentWillUnmount.

ComponentDidMount → it will render after first onwards

ComponentDidUpdate → it will render after every update re-render

Episode:- 8
Assignment

Page No.
Date

SAGAR

Q) How do you create Nested Routes react-router-dom configuration.

For this first of all we import {CreateBrowser, RouterProvider, Outlet} from 'react-router-dom' (after installing npm install react-router-dom in this)

and Now we will create configuration

```
const anyVariableName = createBrowserRouter([
    {path: "/",
     element: <AppLayout />,
     errorElement: <Error />,
    }
])
```

if u r including this → Let appRouter

to render which Compo. → element: <AppLayout />,

this is for Error → errorElement: <Error />,

we can also use children

```
{path: "/about",
 element: <About />,
 children: [
    {path: "profile",
     element: <Profile />,},
 ]
},
```

This part will render when URL is About/Profile

Outlet - we use this as it is an Component which fill up the configuration

We put <Outlet /> & how our requirement as to render the about Configuration

for

```
const AppLayout = () => {
    return (
        <Header />
        <Outlet />
        <Footer />
    );
}
```

here So, it will be interacting Header & Footer.

Note → outlet always created in parent (children)

RouterProvider is component which is coming from ReactRouterDom
Now to provide Route to rootrender
() we use RouterProvider & Now this will take this configuration router as root.render(<RouterProvider router={appRouter} />)

② what is the order of life cycle method calls in Class Based Components ?

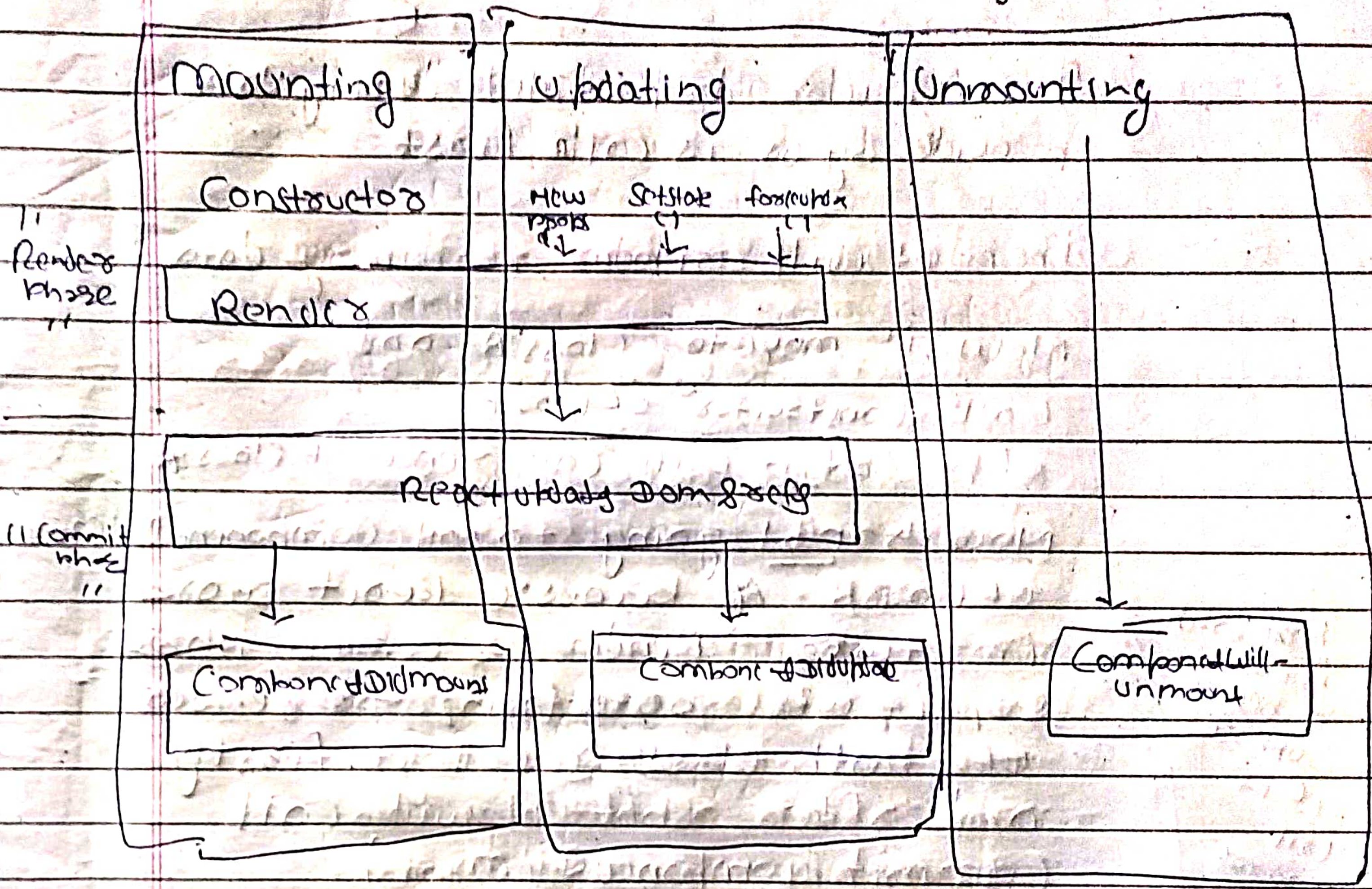
→ Firstof All

```
class Profile extends React.Component {

}
```

* In this call firstly Constructor call, then render call then ComponentDidMount but when their is more than class

Component as child then Scenario is Changed ~~[illegible]~~ their is 3-cycle



Lets me explain these all by one ex-
Suppose we have class

class Ankit extends React.Component {

~~[illegible]~~ send

1st — constructor { }

3rd — componentDidMount { } API call }

render {

2nd here In render whole call other ~~render two~~ other class

Let it be (A) — <userclass name='Ankit' />

Let it be (B) — <userclass name='Ankit' /> }

what happen is that In first phase ~~Constructor~~ mounting

- In Constructor do provided dummy data provide by us is calle first
- then it will render the dummy data

Now it move to ClassA and Call Constructor of ClassA after that it will call render of ClassA Now its not going to call ComponentDidMount of ClassA because React work in two Render & Commit phase So, it will both (join) the render of both Class & call it firstly and after that it will call ~~Commit~~ ComponentDidMount)

Now it goint to call Constructor & then call render of ClassB then after that it will call ClassA ComponentDidMount then ClassB componentDidMount and last post ComponentDidMount

here both is done because Suppose if it call ~~render~~ first componentDidMount of 1st and after that it render 2nd one and call 2nd componentdidmount then it will make it slow this is why it both render.

Mounting { Constructor (dummy)
Render (dummy)
(HTML) dummy
ComponentDidMount
(API call)
(this.setState)

Updating { render (API)
HTML (API data)
ComponentDidUpdate

Page No.
Date

Mounting [→ As soon as ~~it~~ class loaded
constructor will call and after
this render happen & ~~react~~
now in commit phase react
create ~~update~~ dom by dummy data

Now componentDidMount call &
in that componentDidMount API is
called ~~by~~ (as it is best place to call API)

See diagram

Updating [& in componentDidMount we have this.setState
which is call & it will update state
variable. Now this happen react will
render once again and now react
will update dom with new value coming
from api and now it will

call componentDidUpdate ~~it~~ and
lastly unmount stage

Unmounting { componentWillUnmount → (remove from UI) when we
go to new page it will call.

previous ...

above [Code] Class ~~Anka~~ {
constructor () {
this.state = {
userInfo : {
name : Dummy
os : JS
}
} dummy data before update
}
async componentDidMount () {
(API call) fetch, json
~~this.setState {~~
userInfo : json } }
}

render() {
return

(div) JSX {ele / non} data from api)

}

Q> why do we use ComponentDidmount

→ In React life cycle it is call after constructor & render : this will trigger when we visit page only once and it is best place to call api.

it is particlary use for performing task that should occur only once such as fetching data from API, Subscribing to event

→ it is important to not that ComponentDidmount() is not call when the Component is rerendered unless the Component is unmounted & mounted again. and if we need to perform tasks when Component updates or receive new props we use other lifcycle ComponentDidUpdate().

Page No.
Date

Q. Why do we use ComponentWillUnmount? Explain with example.

→ it will triggers when we leave page or close it.

ComponentWillUnmount() {

clearInterval(variable name or this.time)

}

OR - CleaningUpEventListeners → it imp to remove the event listeners to avoid memory leaks.

Q. Why do we use super(props) in constructor

→ In JavaScript the super() keyword is used to call the constructor of parent class it is necessary to call 'super(props)' in constructor.

- Accessing this.props - by calling super(props) the props parameter is passed to the parent class constructor. This allow you to access this.props within constructor or without calling 'super(props)' this.props would be undefind in constructor.

Page No
Date

2- Initializing state :- In many cases, the Constructor is used to initialize component state by calling 'super(props)' we ensure that parent class constructor is called first, which set up the component internal state properly. This enable you to access this state and initialize it with default value or value from passed from props.

2 Why can't we have the callback function of useEffect async?

Similar to async useEffect can't use why?

In react 'useEffect' hook is designed to work with synchronous function & does not directly support using async function as callback functions. This is because useEffect hook expects the callback function to either return cleanup function or not return anything at all.

But we can use a workaround to use async function within the useEffect hook. but remember that the cleanup function will be synchronous and not rely on any asynchronous operation

```
useEffect(() => {
  const fetchData = async() => {
    try {
      const res = await fetch('https://...')
      const data = await res.json();
      setState(data);
    }
    fetchData();
    return () => {
      // Cancel - perform cleanup if any
    }
  }, []);
```