(12/3/23)
11/3/23

(L-1)

[React]

Date: 11
P. No:
Rupesh Patil

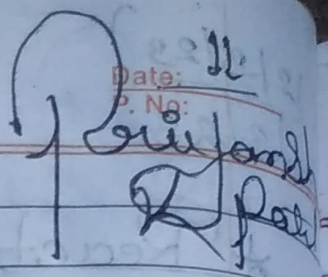
* React
→ JS library which is used to create UI.

* Why React
→ React is based on Declarative approach where JS is Imperative approach.

* SPA
→ Single Page Application
→ Only one HTML Page and changes
→ occur dynamically.
React controls UI.

* Components
→ Component is Reusable.
→ It is a piece/block of code.
→ React is all about Component
→ React is component based architecture.
→ Increase Reusability.
→ Custom HTML Element
* App.js → First Component
* index.js → Execute first.

* Setup of React App
Step 1 → Install Node JS
Step 2 → npx create-react-app appname
Step 3 → cd appname
Step 4 → npm start



* Create Component

→ Create JS File
→ Return JSX.
→ Import in another file
→ Export Component.

* className

'class' is reserved word. So that's why we used className.

* Props

→ It refers to the arguinment or data passed from parent to child Compo

→ It is way to sharing info b/w diff Components.

→ for pass props, simply include attributes whe rendering the child component.

Ex→
```
<parent>
  <child name="priyansh" />
</parent>
```

Here props pass parent to child compone

```
Const Child = (props) => {
    return (
        <div>
        <h1> Hello, {props.name} </h1>
        </div>
    )
}
```

Child components takes "props" object as an argument and use the 'name'.

14/9/23 React - II

Date:
P No:

Nirvansh Patel

* For event handelling we don't give parenthesis () on func. name.
  Ex - onclick = {clickHandler}

* States
  State return, inital value and function.

  changing function
  ↓
  Const [title, setTitle] = useState(props.title);
  ↑ initial value

→ "State" is a JavaScript object that represent the current state of a component.

→ It re-render of the component when that data changes

* How to pass data from child to parent Component
  We can pass data from a child component to its parent component by using callback function as a props -

Steps
① Define function in parent component that will receive data from the child component

② Pass the function as a props to child component

③ In child, call the function and pass the data as an argument

Note → Use a callback function as a props instead of directly modifying the parent component's state from the child components.

18/03/22 [Reacct - III]

Date: !!
P. No:

* UseEffect :-
The useEffect Hook allow you to perform side effects in your component.
ex :- updating DOM, fetching data, timers

Variation 1 → pass empty array for single render page
useEffect(() => { — }, [])

Variation 2
useEffect(() => { — });
→ render everytime when change occur

Variation 3
when dependency change then it render.
useEffect(() => { — }, [name]);
when name change then it render

Variation 4
useEffect(() => {
[ ] ← बाद में होगा
return () => {
[ ] ← पहले execute होगा
}
}

* Component Mount → When Component Render

* Component Unmount → When Remove from DOM

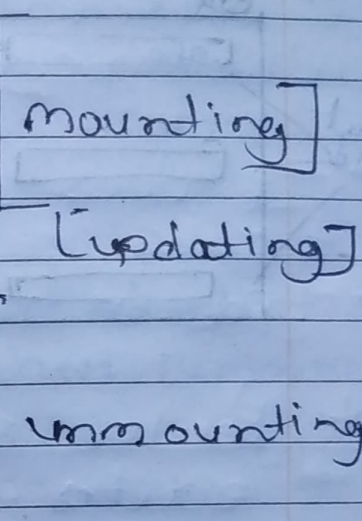
!!

Date:
P. No:

Ravi Ranjan Singh Rahul

* Life Cycle of Component.

→ Now, we used useEffect instead of life cycle.



* Mounting

→ When component first created and added to Dom
→ useEffect hook with an empty dependency array [] will run only once.
→ It is good place to add - event listener, API.

* Updating

→ Component Re-render due to changes in its state or props
→ useEffect with dependency will run whenever dependency change
→ It is good place to add - DOM.

* Unmounting

→ Occur when component is removed from DOM.
→ Used for clean up any resource that were created during lifecycle of component. Such as - Event Listener, timer

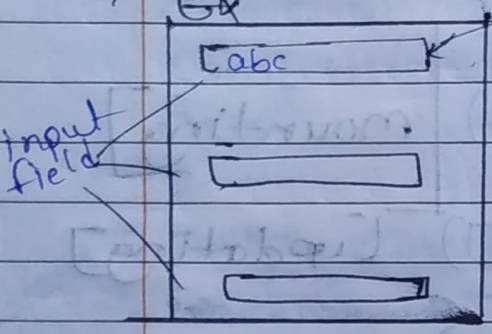
20/03/23 React VI

Date: 11
P. No:

* Controlled Component

→ Maintain state inside Component

→ Ex



changeHandler ()
{
setFormData()
}

by this render again JSX
So we give value in input

```
<input
  value = {formData.Name}
/>
```

21/03/23 [React - VII]

Date: 11
Page No:

Rutuja Patil

* React Router

→ Navigate multiple pages without refreshing page.

→ npm Install react-router-dom

* BrowserRouter (Router)

→ It is the parent component that is used to store all the other component.

* Routes

→ All Route contains in Routes

* Route

→ Check current URL and display the component associated with exact path

* React Router

→ It is a package that allow you to add routing in React application.

→ Provides way to navigate b/w different Component based on URL of the app

→ Define Routes and render specific Component based on URL.

→ Using it create Single Page Application with multiple view, without having to reload entire page

Date: 11
P. No:

* Browser Router

→ It is way to create web pages that can change what's shown on the screen without having to reload the entire page

→ It does this by using "HTML5 history API"

→ Faster and more responsive to user

* Routes & Route

→ "routes" refer to the different paths or URLs that you want to handle in application.

→ "route" refer to specific path or URL that you want to map to a specific Component.

* Link

→ Link component is used to create a clickable link that allow users to navigate to different routes/pages within application

→ It is part of "react-router-dom".

→ 'to' props specifies the Path/URL that the link should navigate.

Ex:- <Link to="/about">Click</Link>

Priyansh Patel

* NavLink

→ It is similar to Link.

→ NavLinks adds an "active" class to the rendered element when URL matches the 'to' props of the component.

Ex: <NavLink to="/about">About</NavLink>

* Outlet

Outlet in React is like a placeholder for components to be rendered inside another component. It's a way to ~~rece~~ organize and structure the content of a component by allowing other components to be inserted into specific areas of it.

* useNavigate Hook

useNavigate hook in react is a function that can use to navigate programmatically to a different route / URL within application.

Ex →

```
Const navigate = useNavigate()
<button
    onClick={() => navigate("/about")}
>
    Go to About Page
</button>
```

28/08/22. React Advanced

Date: 11
P. No:

Rutuja Patil

* Context API

It is a way to pass data down the component tree without having to manually pass props through every single component.

* Rules of Context API

→ Creation of Context Object
createContext() & access by itself and all children

→ Define provider Component

→ Consume the context, In any component that needs access to the context data must use 'useContext' hook from react.

* Context API

It is a feature in React that allow you to share data between components without having to pass the data through every level of the component tree as props.

* useLocation Hook

→ useLocation hook in React Router is a hook that provides information about current URL.

Example

→ "https://example.com/my-page?param1=value1&param2=value2"
location.pathname = /my-page
location.search = ?param1=value1&param2=value2

2/4/23 React - Advanced

Date: 11
P. No.:
Gurjansh Patel

* UseSearchParams

→ Access and manipulate query parameters of URL.

→ Return array of two items
(i) object (query Parameter) (ii) function to update those q. parameters

Ex const [searchParams, setSearchParams] = useSearchParams()

→ searchParams.get("q") line retrieves the value of the "q" query parameter from the URLSearchParams Object.

* Redux

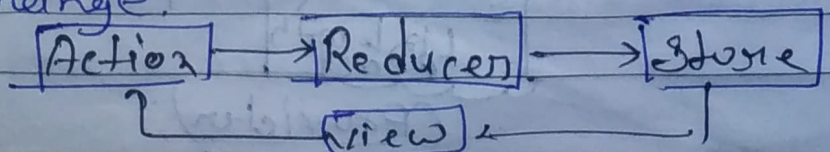
React Redux is a library that helps manage the state of a React application. It simplifies the way data is passed b/w components, making it easier to build complex applications.

Action & Reducer

An action is a plain object that describes the intention to cause change

A reducer is a function that determines changes to an application's state. return the new state and tell store how to do.

It uses the action it receives to determine the change.



*Slice

→ Create Slice by createSlice({ })
- name { name of slice
- initialState { initial state value
- reducers { all the functionality

Must do 2 things
- Export functionality by actions
  - export ——— = counterSlice.actions
    - sliceName
    - function

*Redux Store

Store Create by configureStore.
- export const store = configureStore({
  reducer: {
    counter: counterSlice
  },
});

*Provider

Wrap APP component inside provider

```
<Provider store={store}>
  <App />
</Provider>
```

* Redux Toolkit (MUST DO)

Store

```
→ const store =
    configureStore({
      reducer: {
        counter : counterSlice
        (name in slice)
      }
    })
```

Slice

→ createSlice
- name
- initialState
- reducer

always export
- → export → function (action reducer)
- → export → reducers

Use with React (Provider)