TECH MAJOR

WEB UI

WORKING REACT

WHAT IS REACT

- JS library to build user interfaces (https://react.dev/)
  - Example: Netflix, Facebook
- Created by Facebook
- Widely used for creating web application, particularly SPAs
- React is known for its flexibility, and performance

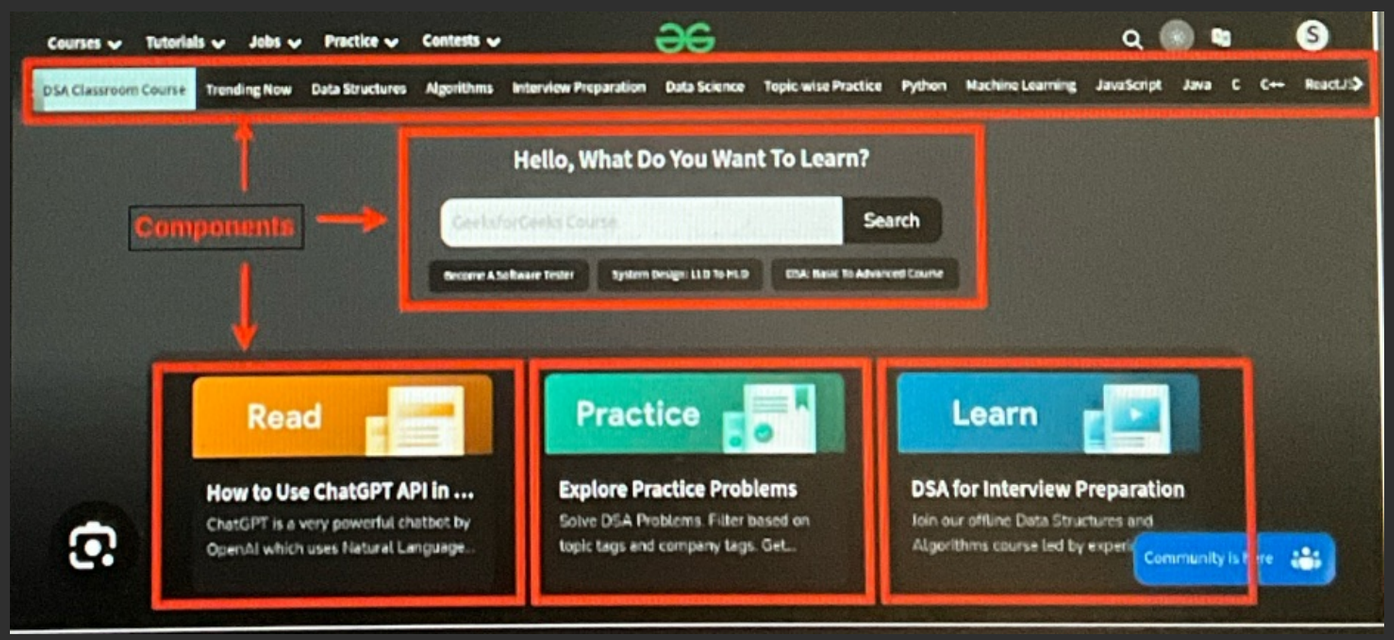
⭐ Single Page application

↳ It's an application which loads one time, & rest everything is done by J.S like routing (Go from one page to another.)

⭐ Component based Architecture

↳ It means that we break down a page into components which are reusable.

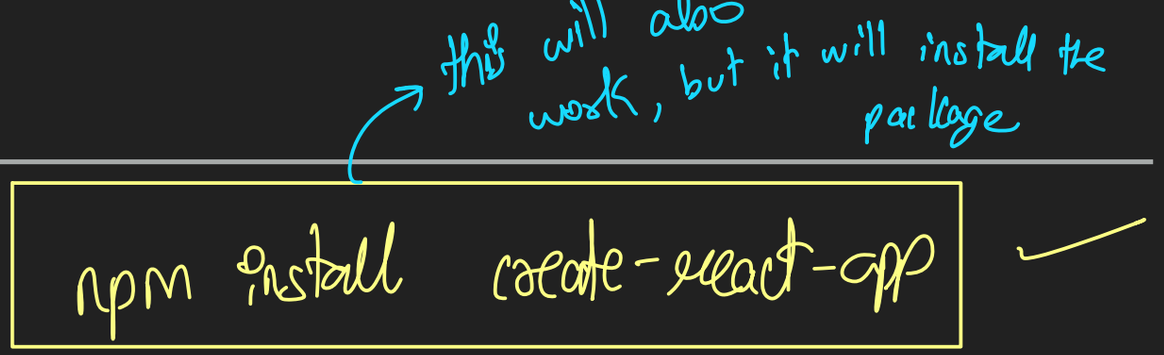
↳ follows write once & use everywhere principle



WHY REACT ?

- Faster development
- Break down UI into reusable components
- UI and logic can be handled together. Not possible with plain JS
- Updates without reloading - SPA
- Much better code organisation
- Declarative syntax (vs imperative way)
- Building Single Page Apps
  - React can be used to build a part of a page OR
  - It can be used to build the complete front end of a Web app with all pages and all UI components
  - Server sends only one HTML page (with JS) to start with
  - React controls everything after that

NPM



- NPM stands for "Node Package Manager"
- It is a package manager for JavaScript
- https://www.npmjs.com/
- Package management
  - A package manager is a software tool that automates the process of installing, upgrading, configuring, and removing software packages on a computer system
  - Dependency Resolution
    - Package manager ensures that the required dependencies are installed and compatible with the software you're installing
  - Repository Management
    - Package managers often interact with repositories, which are collections of software packages.
- NPM has a registry of JS packages
  - Over 1.3 Million packages (according to wikipedia)
- Install node and npm
  - https://docs.npmjs.com/downloading-and-installing-node-js-and-npm
- NPM is an essential tool in the JavaScript ecosystem

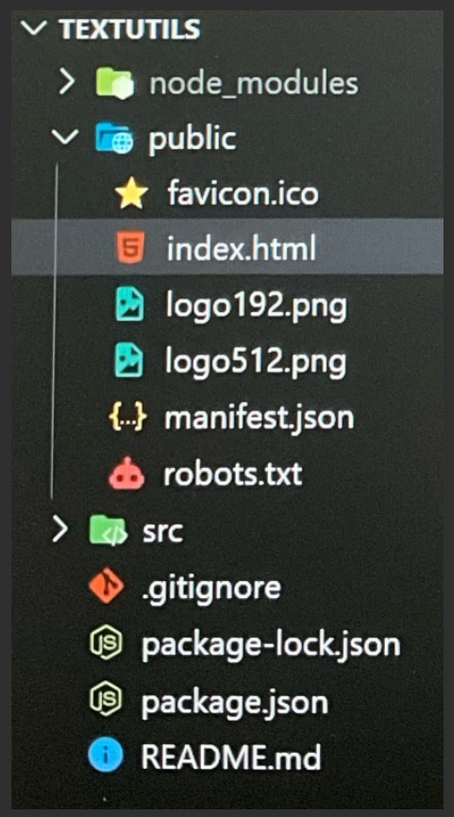
DEVELOPMENT ENVIRONMENT

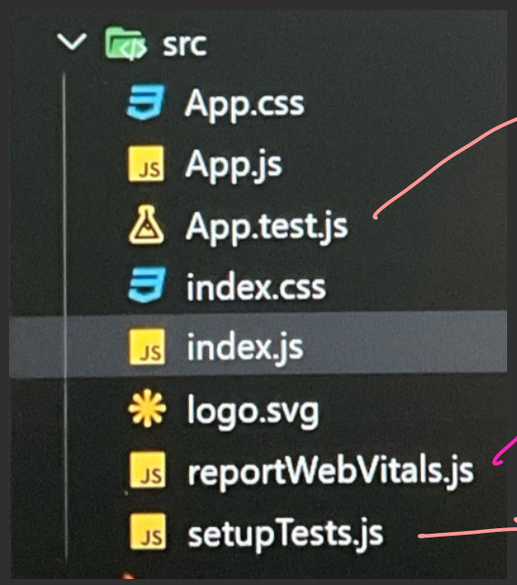
- VS Code - IDE
  - https://code.visualstudio.com/download
- Node.js
  - https://nodejs.org/en/
  - npm (the package manager)
  - npx
- Git - version control
  - https://git-scm.com/downloads

CREATE REACT APP

- Create react app
  - https://create-react-app.dev/docs/getting-started/
  - Create React App is an officially supported way to create single-page React applications
- Make sure you have installed npx (install node from https://nodejs.org/en/)
- npm install
- code walk through in VS Code
  - Index.js
  - Package.json
  - index.html

⭐ Folder structure





here we write all the test cases

loading speed, interactivity, etc

To configure your testing env.

① node_modules → here we have all the installed packages

---

② package.json → contains a list of pkgs which are installed (dependencies)

↓

we put node_mod" in .gitignore, so via this we are able to install all pkgs.

---

③ package-lock → ensures exact version & one pkg is dependent on what other pkg also, it has its list.

---

④ index.js ⇒ entry point

⑤ App.js ⇒ we will change code inside app.js & index.js will render it.

manifest file

↓

use to describe our application in mobile phone

Robots.txt → use to prevent the search engine & bots to crawl up their sites

↳ basically it tells —ti— which pages or files the crawler can or can't request from your site.

App.js | index.js

src > index.js

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3   import './index.css';
4   import App from './App';
5   import reportWebVitals from './reportWebVitals';
6
7   ReactDOM.render(
8     <React.StrictMode>
9       <App />
10    </React.StrictMode>,
11    document.getElementById('root')
12  );
13
14  // If you want to start measuring performance in your app, pass a function
15  // to log results (for example: reportWebVitals(console.log))
16  // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17  reportWebVitals();
18
```

Via strictMode

① Its a tool for highlighting potential problems

② Gives out warning

③ Identifies side effects

④ Detect unsafe practices, etc.

JSX

JSX = JS + HTML

{name's} → use brackets & write JS
{console.log("H")}

- Javascript extended
  - https://react.dev/learn/writing-markup-with-jsx
- Describe what the UI should look like
- JSX produces React “elements”
- Its HTML code inside JS
- React transforms JSX to HTML and JS
  - Browsers do NOT understand JSX

Unterminated JSX
contents nhi dalne hai
code mai
↓
like
<img> ✗
<img /> ✓
<input> ✗
<input /> ✓

App.js M

src > App.js > App

```
1   import logo from './logo.svg';
2   import './App.css';
3
4   function App() {
5     return (
6       <div className="App">
7         <header className="App-header">
8           <img src={logo} className="App-logo" alt="logo" />
9           <p>
10            Edit <code>src/App.js</code> and save to reload.
11          </p>
12          <a h
13            className="App-link"
14            href="https://reactjs.org"
15            target="_blank"
16            rel="noopener noreferrer"
17          >
18            Learn React With Harry
19          </a>
20        </header>
21      </div>
22    );
23  }
24
25  export default App;
```

Now, the filename is .js & we are writing HTML inside what is this ?

↓

so basically this is JSX

JSX = JS + HTML

here we write JS inside `{}` brackets.

In react you have to return a single div always

or

you can wrap in JSX fragment `<></>`

WHAT IS A REACT COMPONENT

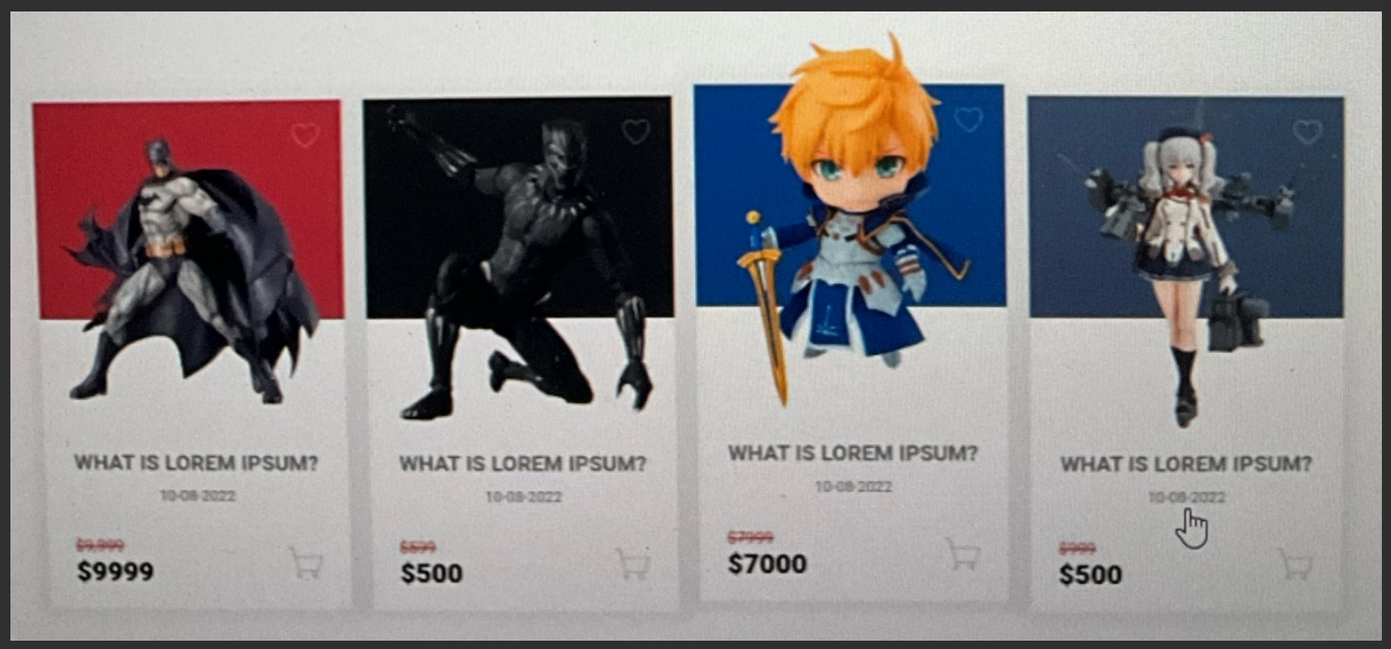
- https://react.dev/learn/your-first-component
- Components let you split the UI into independent, reusable pieces
- A Component describes a part of the UI and may also contains the logic associated with it
- They accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen
- Components can be implemented as classes or functions
- UI is broken down into multiple components
- Each component is implemented in isolation

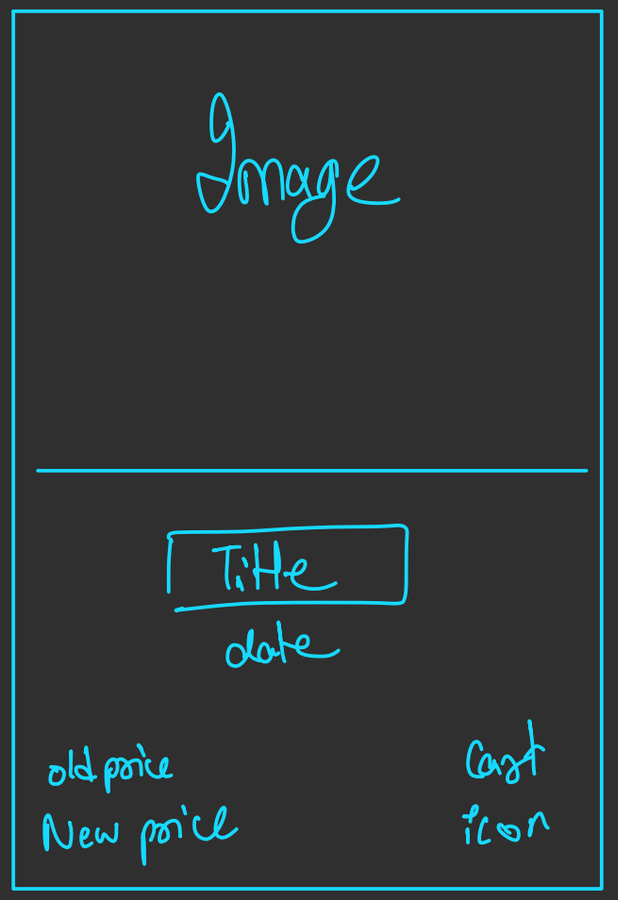
↳ By individual or grp of people

You can also think a component as a template.

Eg:-

import './card.css'





often making this template in HTML (JSX inside component) you can give data to each component via props

FUNCTION COMPO

- React component is a JavaScript function that yo can sprinkle with markup
- 1. Export component
- 2. Define function
- 3. Return markup
- 4. Use component

```
1  export default function Books() {
2    return (
3      <ul>
4        <li>Eloquent Javascript</li>
5        <li>Javascript: Definitive guide</li>
6        <li>Javascript: The good parts</li>
7        <li>You don't know JS: Scope and closures</li>
8      </ul>
9    )
10 }
```

function should return JSX & react will convert to JS & HTML

```
1  export default function ListBooks() {
2    return (
3      <Books />
4    );
5  }
```

CLASS COMPONE

- React lets you define components as classes
- Must define the render() method
- It returns the markup

```
class Books extends React.Component {
  render() {
    return (
      <ul>
        <li>Eloquent Javascript</li>
        <li>Javascript: Definitive guide</li>
        <li>Javascript: The good parts</li>
        <li>You don't know JS: Scope and closures</li>
      </ul>
    );
  }
}
```

```
class ListBooks extends React.Component {
  render() {
    return (
      <Books/>
    )
  }
}
```

PROPS → short for "properties"

- React components use props to communicate with each other
- Every parent component can pass some information to its child components
- Read-only
- Makes a component dynamic and customisable
- Any number of props
- Props can be of any type. Any js value can be passed
  - Objects, arrays, functions
- https://react.dev/learn/passing-props-to-a-component

Eg :- `<Navbar title="textutils" />` → then inside component

new property added

should be read only ← can't change props inside func^n

```
export default function
Navbar (props) {
  return (<>
    <h1>{props.title}</h1>
  </>)
}
```

⭐ Prop Types → used to avoid sending wrong data

Eg:-

```
import PropTypes from 'prop-types'

export default function Navbar (props) {
...
```

3

```
Navbar.propTypes = {title : PropTypes.string,
                    aboutText : —u—
}
```

if the parent component sends a number, then it will throw an error

You here you can also set some mandatory props which the parent has to send.

throws error

```
Navbar.propTypes = {
title : PropTypes.string.isRequired,
aboutText : —u—
}
```

But if you set default props & parent component don't send it, then 'no error'

Just like input field in HTML, here parent component has to send those props.

✡ Default Props

↳ you can set some default values to certain props, in case it is not an mandatory field which the parent component has to send.

Eg:-

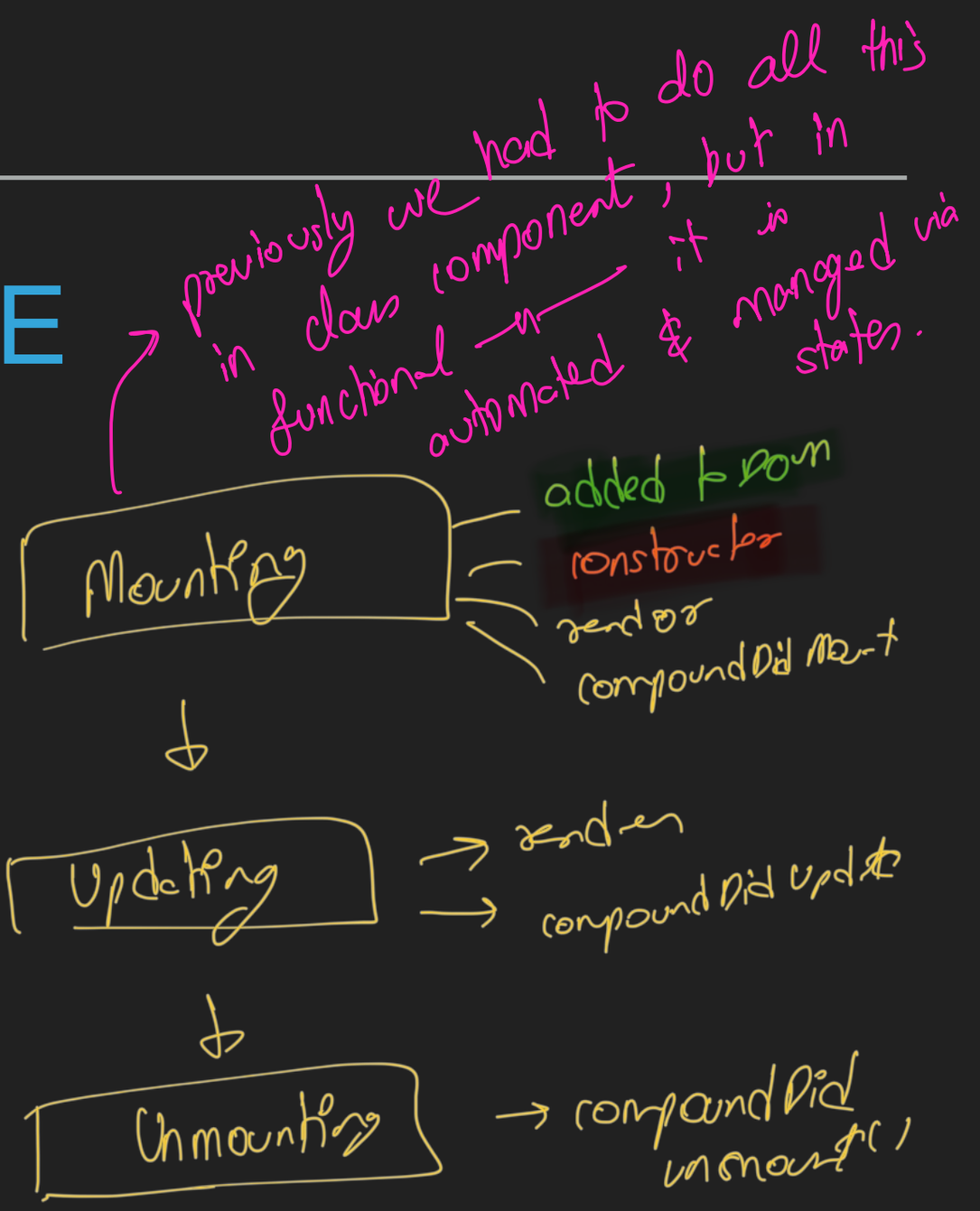
```
Navbar.defaultProps = {
    title: 'Set title here ...',
    aboutText: 'About text here ...'
}
```

default props can fulfill the requirement
↓
& no errors will be shown for required props
↓
since they got their values.

Good practice to write
↳ default props
↳ prop types

COMPONENT LIFECYCLE

- Each component undergoes these phases in its lifecycle
- 1. Mounting - Component is added to the DOM
  - constructor()
  - render()
  - componentDidMount()
- 2. Updating - Component is updated
  - render()
  - componentDidUpdate()
    - This is where new data can be fetched based on the new state
    - If this changes state, it has to be inside a conditional statement
      - Example: if(prevState.id != this.state.id) { make API call }
- 3. Unmounting
  - componentDidUnmount()
- https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/



RE-RENDERING COMPONENTS

- A component can be rendered multiple times
- A component is re-rendered when
  - State of the component changes
  - Props change
  - Parent component is re-rendered
  - A few other cases
- What happens
  - Component redraws the UI with the changed state or props

⭐ States → "Awaster"

↳ Ek variable hai & jab woo variable update hoo, tho jaha jaha woo variable use hua tha, woo bhi re-renders hojaye. → This is done without reloading the page

↳ This belongs to a component

∴ you should use "usestate" hooks inside a functional component.

Eg:-

```
import {useState} from 'react'

export default function TextJoom(props) {

    const [text, setText] = useState("");

    setText("Enter Text here");

    ...

}
```

→ you can update only like this `text="abc"` ✗

→ you will get error

State variables

↓

React will keep a watch on this, by default react sabpe watch nhi rakhi hai → using this watch it will re render the part when state is changed.

Eg:-

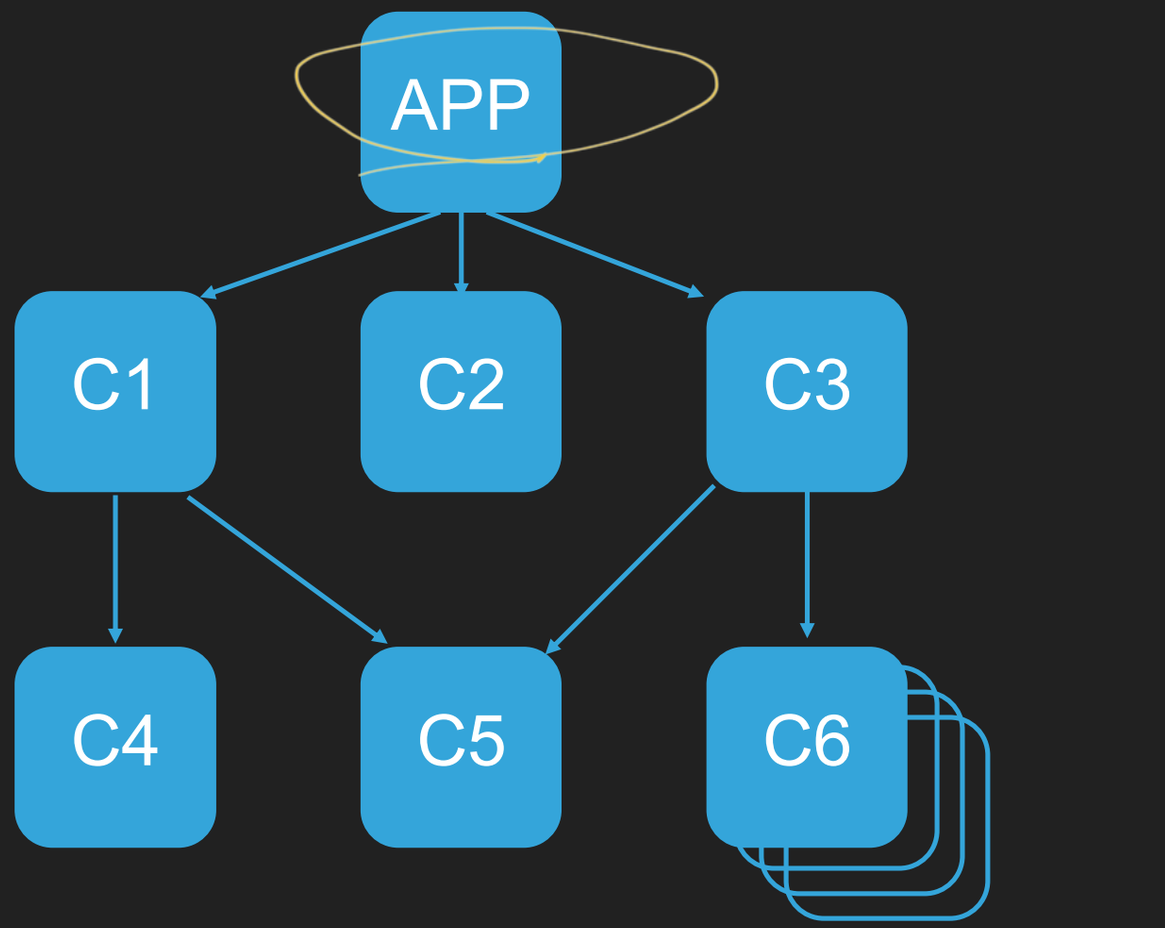
components > JS TextForm.js > TextForm

```
import React, {useState} from 'react'


export default function TextForm(props) {
    const handleUpClick = ()=>{
        console.log("Uppercase was clicked" +  text);
        let newText = text.toUpperCase();
        setText(newText)
    }

    const handleOnChange = (event)=>{
        console.log("On change");
        setText(event.target.value)
    }

    const [text, setText] = useState('Enter text here2');
    // text = "new text"; // Wrong way to change the state
    // setText("new text"); // Correct way to change the state
    return (
        <div>
            <h1>{props.heading}</h1>
            <div class="mb-3">
            <textarea class="form-control" value={text} onChange={handleOnChange} id="myBox" rows="8"></textarea>
            </div>
            <button class="btn btn-primary" onClick={handleUpClick}>Convert to Uppercase</button>
        </div>
    )
}
```

Is variable (state)

COMPONENT TREE

- App Component is the root of the component tree
- C1 is instantiated only when its encountered in App's render()
- C1 is inserted in the DOM
- C1's render is not called again until:
  - Input props from App change
  - State of C1 changes



Same component, multiple instances

WHAT IS REACT ROUTER?

- How to share links to certain parts of the App? Say, a product page.
- Serving different html pages for different urls is slow (Network + rendering delay)
- Client side routing
- Url changes when content changes
- React changes the content without fetching a new html
- React router handles navigation and updates the UI accordingly

REACT ROUTER PACKAGE

- React Router (https://reactrouter.com/)
- Installation
  - *npm install react-router-dom@6*
- Latest version is 6
- Some differences from version 5
- Conditionally renders components
  - techmajor.io/ - Component A
  - techmajor.io/courses - Component B
- Matches Urls to routes
- Renders UI based on route matched
- Manipulate history stack

⭐ React router

↳ 2) why do we use it?

↳ what's the problem with anchor tag?

↓

Every time we click on tag it causes <u>refresh</u> of whole page

but we want to load the content without <u>refresh</u>

```
const Navbar = () => {
  return (
    <div>
      <nav>
        <a href="/"><li>Home</li></a>
        <a href="/"><li>About</li></a>
        <a href="/"><li>Contact</li></a>
      </nav>
```

→

npm i react-router-dom

ROUTER COMPONENTS

- <BrowserRouter> - Interface for running router in a web browser
- <Routes>
  - Url to route matching. Best match is returned
  - In Version 5, <Switch> was used. It used to return the first available match
- <Route>
  - Child of <Routes>
  - If path matches the url, the corresponding element is rendered
  - Url to component mapping
  - Nested routes are allowed
- Link
  - an element that lets the user navigate to another page by clicking on it.
  - Prevents page refresh
- useNavigate hook
- useParams hook

∵ we want this globally ←

under app.js file

① create BrowserRouter

↓

takes in array of obj

to create router

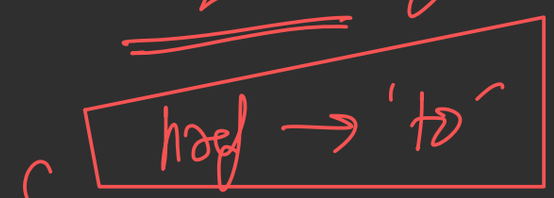
Eg:- [{path: '/', element: <Home/>}]

Eg :-

```jsx
import * as React from "react";
import * as ReactDOM from "react-dom/client";
import {
  createBrowserRouter,
  RouterProvider,
} from "react-router-dom";
import "./index.css";

const router = createBrowserRouter([
  {
    path: "/",
    element: <div>Hello world!</div>,
  },
]);

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <RouterProvider router={router} />
  </React.StrictMode>
);
```

If you did all this & then click on the anchor link, then also, the page will reload we don't want it to reload

∴ replace anchor with Link tag,



runtime mai ye anchor tag mai hi convert hota hai

```jsx
function Navbar() {
    return (
        <div className="nav">
            <Link to="/" className="nav-item">
                Home
            </Link>

            <Link to="/contact" className="nav-item">
                Contact
            </Link>
            <Link to="/product" className="nav-item">
                Product
            </Link>
        </div>
    );
```

⭐ Passing Parameters in URL

Use Params Hooks

1

```
const router = createBrowserRouter([
    { path: "/", element: <Home /> },
    { path: "/product/:productName", element: <Product /> },
    { path: "/contact/", element: <Contact /> },
]);
```

2

```
import Navbar from "../components/Navbar";
import { useParams } from "react-router-dom";

Tabnine | Edit | Test | Explain | Document | Ask
function Product() {
    const params = useParams();
    return (
        <div>
            <Navbar />
            <div className="container">
                <h1>Product: {params.productName}</h1>
            </div>
        </div>
    );
}
```

⑧ Going to a route via an event

useNavigate Hook

```jsx
import React from "react";
import { useNavigate } from "react-router-dom";

Tabnine | Edit | Test | Explain | Document | Ask
function ProductCard(props) {
    const navigate = useNavigate();
    function goToProduct() {
        navigate("/product/" + props.title);
    }
    return (
        <div className="product-card">
            <h2 className="title">{props.title}</h2>
            <p className="desc">
                Lorem ipsum dolor sit amet consectetur adipisicing elit.
                Impedit id doloremque omnis, aperiam in temporibus! Facilis,
                fuga quo.
            </p>
            <h3 className="price">$500</h3>
            <button className="btn btn-warning" onClick={goToProduct}>
                Add to cart
            </button>
        </div>
    );
}

export default ProductCard;
```

HOOKS

WHAT ARE HOOKS?

- Hooks let you use different React features from your components
- Many built-in hooks
- Important hooks
  - useState
  - useEffect
  - useContext
  - useRef

useLocation

useState HOOK

- State lets components remember information
- Local to the component
- Similar to member variables in a Java class
- Operations on the state
  - Init
  - setState
- A component can have any number of useState hooks

useState HOOK USAGE

- useState takes the initial value as the parameter
  - Can be a value or object
- Returns
  - The state variable
- A function to update that state variable

```
export default function Login() {

  /**
   * State to maintain email and password inputs
  */
  const [loginInfo, setLoginInfo] = useState({email: "", password: ""});

```

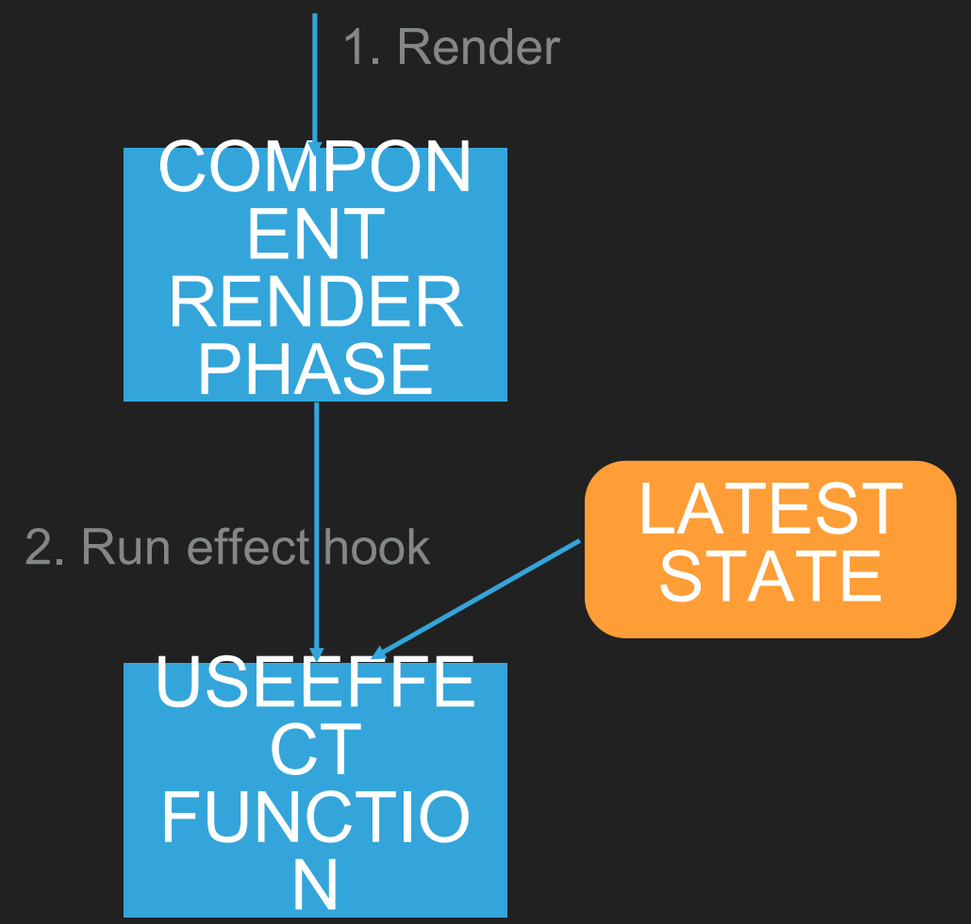
useState HOOK USAGE

- The returned function is called to update the state
- The function gets the previous state as a parameter
- The return value of this function is the new state
- useState is very frequently used with forms to keep track of the value entered

```
/**
 * email change handler
 * */
const emailChangeHandler = (event) => {
  setLoginInfo((prevState) => {
    return {
      ...prevState,
      email: event.target.value
    }
  });
}
```

useEffect() HOOK

- Side effects - Mutations to state, timers, subscriptions, etc
- Side effects to happen outside of the render phase
- Use useEffect() hook instead of the function body.



useEffect SYNTAX AND USAGE

- useEffect(**()=>{}**, [dependencies])
- () => {} - Function that gets executed.
- Can cause side effects
- [dependencies] - Prop or state variables (or any variable)
- Empty dependency list [] makes the effect run only once

FUNCTION
() => {}

This function is executed after the render phase, **IF** the dependencies have changed.

ARRAY OF DEPENDENCIES

The effect function is run only if one or more dependencies change.

useEffect SYNTAX AND USAGE

- Effects are frequently used to connect components to external systems

```
1  function Cart(props) {
2    useEffect(() => {
3      const cartItems = getCartItems(props.userId);
4    },[props.userId]);
5  }
```

WHAT IS CONTEXT?

- Context provides a way to pass data through the component tree without having to pass props down manually at every level
- data that can be considered “global” for a tree of React components
- Example - Authenticated user details
- Every Context object comes with a Provider React component that allows consuming components to subscribe to context changes
- One Provider can be connected to many consumers
- All consumers that are descendants of a Provider will re-render whenever the Provider’s value prop changes

HOW TO CREATE CONTEXT

- createContext creates a context object
- Note that this initial object is NOT used anywhere. Its just a template.
- The context object can also contain functions
- Example - AuthContext
  - keeps track of logged in user info like email, name, token, etc
  - Also has functions to login and logout
- A component stores the context object as a state variable. This component is like a wrapper around the context
- Wrap this component in AuthContext.Provider to specify the value of this context
- The component tree under this Component gets to see the context object.

HOW TO CONSUME CONTEXT

- useContext is a React Hook that lets you read and subscribe to context
- useContext returns the context value for the calling component
- The returned value is always up-to-date.
- React automatically re-renders components that read some context if it changes
- Example - AuthContext
  - Consuming components are re-rendered when the context value changes
  - Component gets the latest value of email, name, etc
  - Consuming component can also invoke functions on the context, if any.

AXIOS

- Axios is a promise-based HTTP Client for node.js and the browser.
- Intercept request and response
- Transform request and response data
- Automatic request body serialization to
  - JSON, multipart, url encoded form
- Automatic JSON data handling in response
- Client side support for protecting against XSRF

WHAT IS VIRTUAL DOM?

- when the app data changes and triggers an update, re-rendering can be expensive.
- When an App's data changes, what is the minimal possible change to the DOM that reflects the new state of the app?
- Reacts way of doing this is called Virtual DOM
- the virtual DOM is a much lighter replica of the actual DOM in the form of objects
- virtual DOM provides a mechanism that allows the actual DOM to compute minimal DOM operations when re-rendering the UI.
- All updates are first done on virtual DOM
- React compares it to a snapshot of the virtual DOM taken just before the update, determines what element was changed, and then updates only that element on the real DOM.
- https://blog.logrocket.com/virtual-dom-react/

THINKING IN REACT

- https://react.dev/learn/thinking-in-react

DEPLOYMENT STEPS

- Test your code 
- Optimise 
- Build the React application 
- Upload production code to server 
- Configure server 

BUILD

- Build - Create a *build* directory with a production build of your app
- Package.json scripts
  - Start
  - Test
  - Npm run build
  - Npm eject
- Run *npm run build*
- Build folder files
- https://create-react-app.dev/docs/deployment/

FIREBASE DEPLOYMENT

- https://console.firebase.google.com/
- Login to firebase and create project
- Go to Build > Hosting
- Deployment using firebase-tools
  - firebase login - logs in the user
  - firebase init - Select project and configure server
  - firebase deploy - deploy build to server
- Share and use url after deployment
- https://tech-major-react-server.web.app/