HTML Steffs

Span → Hello

Padding → space btw boundary and the characters

React

Jsx, class vs className, Static vs dynamic websites, state, components, re-rendering

Facebook was the creator

Why do you need react?

↳ For static websites you Don't

React is just an easier way to write html css and js. Its a new syntax, that under the hood converts to HTML/CSS/JS.

Why React?, Vue Js

JQuery, backbone JS → used to make DOM easier

People realised it's harder to do DOM manipulation the conventional way
There were libraries that came into the picture that made it slightly easy, but still for a very big app it's very hard (JQuery)
Eventually, VueJS/React created a new syntax to do frontends
Under the hood, the react compiler convert your code to HTML/CSS/JS

Some react Jargon

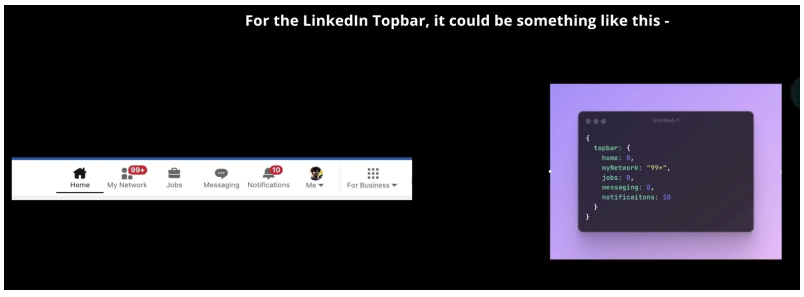
State · Rerendering · Components

State: Represents current state of the app. It represents the dynamic thing in your app (things that change).
eg: value of counter.

↳ Jo bhi JS logic hai eg → notification, counter only in this

↳ How that happens using components

eg



State of Counter App

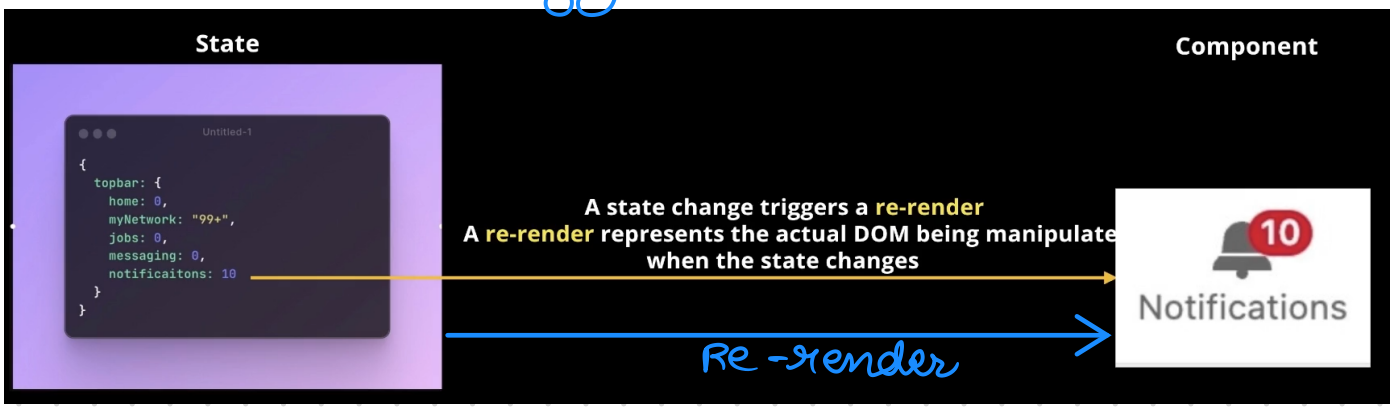
```
{
  count: 1
}
```

Components : Calculate the diff btw old state and new state (DIF calculator)

How a DOM element should render a given state. It is reusable, dynamic, HTML snippet that changes the given state.

↳ Re render

The button is a component. It takes the state (current count) as input and is supposed to render accordingly.



You usually define all components then update state of your app. React takes care of re-rendering your app. ↳ Automatically does DOM Manipulation

Converting value from

↳ 10 → 20

↳ Re rendering

```html
<body>
  <h1>Counter App</h1>
  <button onclick="onButtonPress()" id="btn">Counter 0</button>

  <script>
    function onButtonPress() {
      const currentValue = document.getElementById("btn").innerHTML;
      console.log(currentValue.split(" "));
      const currentCounter = currentValue.split(" ")[1];
      const newCounter = parseInt(currentCounter) + 1;
      document.getElementById("btn").innerHTML = "Counter " + newCounter;
    }
  </script>
</body>
</html>
```

With HTML

<button onclick="onButtonPress()">counter 1 </button>

```html
<body>
    <h1>Counter App</h1>
    <div id="buttonParent"></div>

    <script>
        let state = {
            count: 0,
            color: "white"
        }
        function onButtonPress() {
            state.count++;
            state.color = Math.random() > 0.5 ? "red" : "white";
            buttonComponentReRender();
        }
        function buttonComponent(count) {
            const button = document.createElement("button");
            button.innerHTML = `Counter ${count}`; "counter" + count
            button.setAttribute("onclick", `onButtonPress()`);
            button.style.color = state.color;
            return button;
        }
        function buttonComponentReRender() {
            document.getElementById("buttonParent").innerHTML = "";
            const component = buttonComponent(state.count);
            document.getElementById("buttonParent").appendChild(component);
        }
        buttonComponentReRender();
    </script>
</body>
```

Without HTML
Pure react

button Component = document.createElement("button");
↳ <button></button>

button.innerHTML = "counter" + count;
↳ if count = 1
'counter 1'
↳ <button>counter 1</button>

button.setAttribute("onclick", `onButtonPress()`);
↳ <button onclick="onButtonPress()">counter 1</button>

"So component is the outer thing which gives the structure and state is the inner thing that changes."

Component is like a box and state is like color and number on box that changes

State

React library

Defining the button component

```
<!DOCTYPE html>
<html>

<body>
  <div id="buttonParent">
  </div>
  <script>
    let state = {
      count: 0
    }

    function onButtonPress() {
      state.count++;
      buttonComponentReRender()
    }

    function buttonComponentReRender() {
      document.getElementById("buttonParent").innerHTML = "";
      const component = buttonComponent(state.count);
      document.getElementById("buttonParent").appendChild(component);
    }

    function buttonComponent(count) {
      const button = document.createElement("button");
      button.innerHTML = `Counter ${count}`;
      button.setAttribute("onclick", `onButtonPress()`);
      return button;
    }

    buttonComponentReRender();

  </script>
</body>

</html>
```

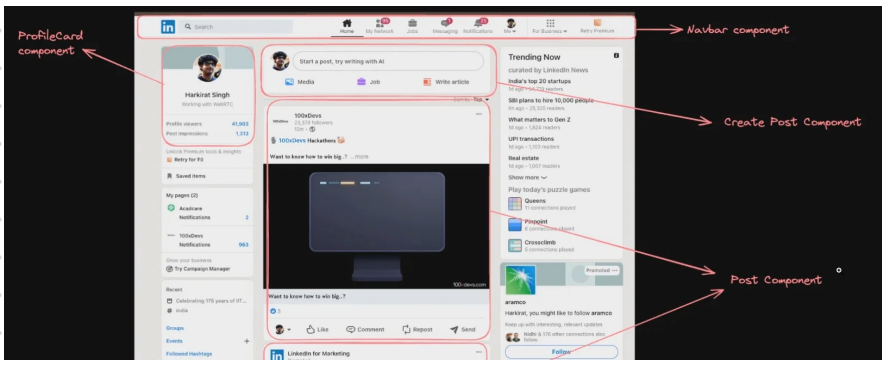
React just re-render

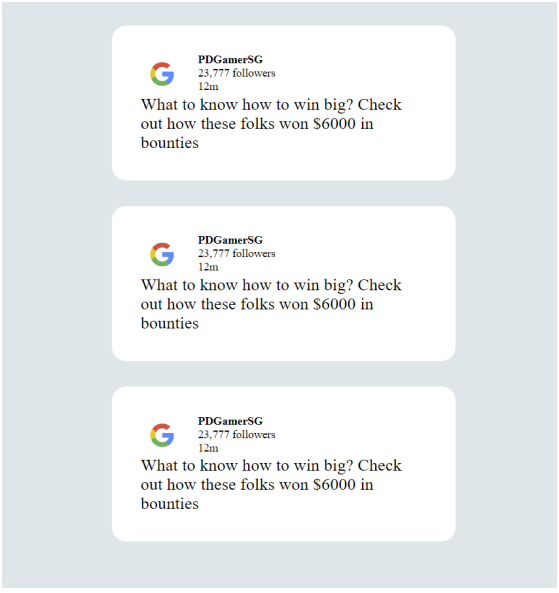
jsx → java script xml

components in jsx → gives xml not html

It is a syntax extension for JS. most commonly used with react, popular JS library for building User Interfaces. JSX allows to write HTML like -code directly within JS. this makes it easier to create and manage the UI in react applications.

components are building blocks of UI, They allow to split UI into indep, reusable pieces, with self contained HTML elements





NO DOM manupilation In React mostly use State Management

App.jsx > Button

```jsx
import React from 'react'

function App() {
  const [count, setCount] = React.useState(0)

  return (
    <div>
      <Button count={count} setCount={setCount}></Button>
    </div>
  )
}

function Button(props) {

  function onButtonClick() {
    props.setCount(count + 1);
  }

  return React.createElement(
    'button',
    { onClick: onButtonClick },
    `Counter ${props.count}`
  );
}

export default App
```

src > App.jsx > ...

```jsx
import React from 'react'

function App() {
  const [count, setCount] = React.useState(0)

  return (
    <div>
      <Button count={count} setCount={setCount}></Button>
    </div>
  )
}

function Button(props) {
  function onButtonClick() {
    props.setCount(props.count + 1);
  }
  return <button onClick={onButtonClick}>Counter {props.count}</button>
}

export default App
```

Two ways of writing Same thing

```jsx
import { useState } from "react";
import "./App.css";

export default function App() {
  const [count, setCount] = useState(0);

  function onClickHandler() {}

  return (
    <div>
      <button id="btn" onClick={onClickHandler}>
        Counter {count}
      </button>
    </div>
  );
}
```

Replit code

const [count, setCount] = useState(0);

```
const value = useState(0);
const count = value[0];
const setCount = value[1];
```

const [count, setCount]

↳ You cannot change the array but you can change the array element

Usestate hook

`useState` is a Hook that lets you add state to functional components. It returns an array with the current state and a function to update it.

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

const Counter = () => {
    const [count, setCount] = useState(0);

    return (
        <div>
            <p>Count: {count}</p>
            <button onClick={() => setCount(count + 1)}>Increment</button>
        </div>
    );
};
```

// changes in raw variables does not reflect in JS.

let count = 0; ✗

↳ Need to be converted into state var which requires (state variable)

React did DOM manupilation

↳ using useState

```
import { useState } from "react";

function App() {
    return <div>
        hi there
        <Counter></Counter>
    </div>
}

function Counter(){
    const [count,setCount] = useState(0);
    function increaseCount(){
        setCount(count+1);
    }
    function decreaseCount(){
        setCount(count-1);
    }
    function resetCount(){
        setCount(0);
    }

    return <div>
        <h1 id = "text">{count}</h1>
        <button onClick={increaseCount}>Increase count</button>
        <button onClick={decreaseCount}>Decrease count</button>
        <button onClick={resetCount}>Reset count</button>
    </div>
}

export default App
```

useEffect

```
function Counter(){
    const [count,setCount] = useState(0);

    console.log("Inside counter component");

    setInterval(function() {
        setCount(count + 1);
    },1000)
    return <div>
        <h1 id = "text">{count}</h1>
    </div>
}
```

→ () going crazy printing Inside counter compo ∞ times and also calling () ∞ times

→ function App(){
return
<counter></counter>

}

If the state variable (count) changes then the it calls the whole counter() again to re render

→ In this everytime new clock is made due to which clock no. goes crazy

hooking into lifecycle events of react.

lifecycle → mounting + re rendering + unmounting

→ It means Pheli baar jabh ghussa

↳ so for our code we just want run the code when it mounts not everytime when it re renders

↳ For that use **use effect**

Conditional rendering

↳ Based on some condn render a component

```
//conditional rendering
function App() {
    let[counterVisible, setCounterVisible] = useState(true);

    useEffect(function() {
        setInterval(function() {
            setCounterVisible(c => !c)
        },5000)
    }, [])
    return <div>
        hi
        {counterVisible && <Counter></Counter>}
        hello
    </div>
}
```

↳ this is true ↑ otherwise nothing

```
function Counter() {
  const [count, setCount] = useState(0);

  console.log("counter");

  // clearInterval
  useEffect(function() {
    console.log("hi there");
    setInterval(function() {
      console.log("from inside the interval");
      setCount(c => c + 1)
    }, 1000);
  }, []) // dependency array, cleanup, fetch inside useEffect

  function increaseCount() {
    setCount(count + 1);
  }
```

Component unmount hogya upar wala code ki wajah se but clock still running.

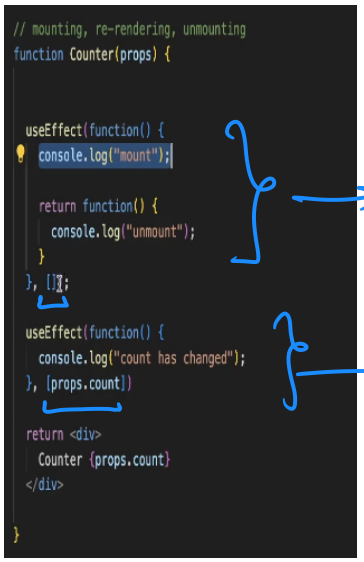
↳ Now cleanup come into picture Jabh mount kiya jabh kuch kiya but unmount par bhi toh kuch karna padega

```
// clearInterval
useEffect(function() {
  console.log("on mount");
  let clock = setInterval(function() {
    console.log("from inside setInterval");
    setCount(c => c + 1)
  }, 1000);

  return function() {
    console.log("on unmount");
    clearInterval(clock)
  }
}, []) // dependency array, cleanup, fetch inside useEffect

function increaseCount() {
  setCount(count + 1);
}
```

→ clear clock from running

so basically the code we write in useEffect is used at the time of mounting and ignored during re-rendering and the () we return in useEffect is called at time of unmounting

Run some logic when count variable changes → Use dependency array



→ when nothing changes runs this

→ when props.count changes run this

```
function Counter(props){
    useEffect(function(){
        console.Log("mount");
        return function(){
            console.Log("unmount");
        }
    },[])
    useEffect(function(){
        console.Log("count has changed");
        return function(){
            console.Log("cleanup inside second effect")
        }
    },[props.count])

    return <div>
        Counter1 {props.count}<br />
        Counter2 {props.count2}<br />

    </div>
```

→ whenever this is called first the unmount () runs and then the upper one which works when counts changes.

Vite → Bundler
↳ convert React code into HTML, CSS, JS

↳ Creates a dev server provide many feature and extremly fast HMR (Hot module replacement)
↳ build command that handles code with Rollup, pre configured to output

main.jsx mai jo app component hai wo render hoga index.html mai
without stopping the process changes happy in final html is due to
↳ (HMR)

J.S → backgroundColor
↳ capital where there is dash
or
"background-color"

To keep element right next to each other
use display: "flex" in style

~~Props~~ Conditional Rendering

```
    <div style={{fontSize : 8 ,marginLeft: 10}}>
        <b>
            {name}
        </b>
        <div>{subtitle}</div>
        {time !==  undefined ? <div style = {{display:"flex"}}>
            <div>{time}</div>
            <img src={"https://png.pngtree.com/png-vector/20211029/
            ourmid/
            pngtree-world-or-earth-vector-illustration-png-image_4014812
            .png"} style ={{width:10, height:10}} />
        </div> : null}
    </div>
</div>
```

State

↳ dynamic part of page

Home, Network → Component

Conditional Rendering

↳ for component re rendering we need a state variable

State var → It is a variable whenever it changes it re renders the page

↳ reechecks has this var changed??

let [isVisible, setIsVisible] = useState(true);

↳ This is two variables which is assigned to variables of array of useState

useState → [true, function]

To update isVisible need to use setIsVisible

```
//the component isnt re-rendering
//because we heavent used a state variable
const ToggleMessage = () => {
    // let isVisible = true;  this is not a state variable
    let [isVisible, setIsVisible] = useState(true);
    function toggle(){
        setIsVisible(!isVisible);
    }

    return (
        <div>
            <button onClick = {toggle}>
                Toggle Message
            </button>
            {isVisible && <p>This message is conditionally rendered!</p>}
        </div>
    );
};
```

↳ This changes value

↳ This function is called again and again

```
const ToggleMessage = () => {
    let [notificationCount, setNotificationCount] = useState(0);
    function increment(){
        setNotificationCount(notificationCount+1);
        // notificationCount +=1;  this will not work
    }

    return (
        <div>
            <button onClick = {increment}>
                Increase Count
            </button>
            {notificationCount}
        </div>
    );
};
```

↳ This will not work

variable going up by 1 but react is not re-rendering

```
return (
    <div style={{background: "#dfe6e9" , height: "100vh"}}>
        <button onClick = {addPost}>Add post</button>
        <div style={{display: "flex", justifyContent: "center"}}>
            <div>
                {[<PostComponent
                name  = {"PD"}
                subtitle = {"11000 followers"}
                time = {"2m ago"}
                image = {"https://t4.ftcdn.net/jpg/03/47/46/83/
                360_F_347468387_ngBE1hQcqRLRVh8d3UDi3YrKNnhfT6V7.jpg"}
                description={"What to know how to win big? Check out how
                these folks won $1000 in bounties?"}
                />,
                <PostComponent
                name  = {"PD"}
                subtitle = {"11000 followers"}
                time = {"2m ago"}
                image = {"https://t4.ftcdn.net/jpg/03/47/46/83/
                360_F_347468387_ngBE1hQcqRLRVh8d3UDi3YrKNnhfT6V7.jpg"}
                description={"What to know how to win big? Check out how
                these folks won $1000 in bounties?"}
                />]}
            </div>
        </div>
    </div>
)
```

To do this print array of posts use

MAP

```
function addPost(){
    setPosts([...posts,{
        name: "PalLab",
        subtitle: "1000 followers",
        time : "2m ago",
        image: "https://t4.ftcdn.net/jpg/03/47/46/83/
        360_F_347468387_ngBE1hQcqRLRVh8d3UDi3YrKNnhfT6V7.jpg",
        description: "What to know how to win big? Check out how these
        folks won $1000 in bounties?",
    }])
}
```

→ everything from older array + the next data into the array (spread operator)

eg

```
let x = [1,2,3]
[...x, 4]
```

→ [1,2,3,4]

```
[4,...x]
```

→ [4,1,2,3]

→ state variable

```
const [posts, setPosts]
    = useState([]);
```

→ bcoz state variable is a array

```
import { useState } from "react";
import { PostComponent } from "./Post";

function App() {
    const [posts,setPosts] = useState([]);

    const postComponents = posts.map(post => <PostComponent
        name  = {post.name}
        subtitle = {post.subtitle}
        time = {post.title}
        image = {post.image}
        description={post.description}
    />)

    function addPost(){
        setPosts([...posts,{
            name: "PalLab",
            subtitle: "1000 followers",
            time : "2m ago",
            image: "https://t4.ftcdn.net/jpg/03/47/46/83/
            360_F_347468387_ngBE1hQcqRLRVh8d3UDi3YrKNnhfT6V7.jpg",
            description: "What to know how to win big? Check out how these
            folks won $1000 in bounties?",
        }])
    }

    return (
        <div style={{background: "#dfe6e9" , height: "100vh"}}>
            <button onClick = {addPost}>Add post</button>
            <div style={{display: "flex", justifyContent: "center"}}>
                <div>
                    {postComponents}
                </div>
            </div>
        </div>
    )
};
```

click Add posts new posts are added

Use effect

↳ To fix the crazyness

↳ useEffect codes works only when the first time the code gets mounted

```
import { useEffect, useState } from "react";

function App() {
    const [count,setCount] = useState(1);
    function increaseCount(){
        setCount(count+1);
    }
    useEffect(function(){
        console.log("above setinterval");
        setInterval(increaseCount,1000);
    }, [])

    return <div>
        {count}
    </div>
}
export default App
```

The code help to run the clock to run once

If we remove useffect then ∞ clocks. Code goes brrrr.....

↳ Count hardcoded to one

If you want to use state variable inside useeffect than add the state variable inside dependency array

↳ Therfore bug

```
function increaseCount() {
  setCount(function(currentValue) {
    return currentValue + 1;
  });

  setCount(currentValue => currentValue + 1);


  function (a, b) {
    return a + b;
  }

  (a, b) => a + b
```

→ Arrow function

Dependency Array

↳ Anytime the value of count changes the function will run. If array is empty it will only run on mount

```
useEffect(function() {
  console.log("above setinterval")
  setInterval(increaseCount, 1000);
}, []) // this effect will run on mount, because
the array is empty

useEffect(function() {
  console.log("the count has been updated to " +
  count);
}, [count]);
```

} Run as the count changes

onClick
↳ requires a ( )

Todo Code with a backend

```
import { useEffect, useState } from "react";
function App() {
    const [currentTab, setCurrentTab] = useState(1);
    const [tabData, setTabData] = useState({});
    const [loading, setLoading] = useState(true);

    useEffect(function() {
        setLoading(true);
        fetch("https://jsonplaceholder.typicode.com/todos/" + currentTab)
            .then(async res =>{
                const json = await res.json();
                setTabData(json);
                setLoading(false);
            });
    }, [currentTab])
    return <div>
        <button onClick ={function() {
            setCurrentTab(1);
        }} style= {{color: currentTab == 1 ? "red" :"black"}}>Todo #1</button>
        <button onClick ={function() {
            setCurrentTab(2);
        }} style= {{color: currentTab == 2 ? "red" :"black"}}>Todo #2</button>
        <button onClick ={function() {
            setCurrentTab(3);
        }} style= {{color: currentTab == 3 ? "red" :"black"}}>Todo #3</button>
        <button onClick ={function() {
            setCurrentTab(4);
        }} style= {{color: currentTab == 4 ? "red" :"black"}}>Todo #4</button>
        <br />
        {loading ? "Loading..." : tabData.title}
    </div>
}
export default App
```

cleaning the code

cleanup ( ) used bcoz without this code works but if the function gets unmounted the function does not stops running

```
  const [showTimer, setShowTimer] = useState(true);

  useEffect(() => {
    setInterval(function() {
      setShowTimer(currentValue => !currentValue);
    }, 5000)
  }, [])

    return <div>
      {showTimer && <Timer />}
    </div>
}

const Timer = function() {
    const [seconds, setSeconds] = useState(0);

    useEffect(function() {
        setInterval(function() {
          console.log("from inside clock")
          setSeconds(prev => prev + 1);
        }, 1000);
        // cleanup functions

    }, []);

    return <div>{seconds} seconds elapsed</div>;
};
```

this ( ) still running when timer is removed

```
const Timer = function() {
    const [seconds, setSeconds] = useState(0);

    useEffect(function() {
        let clock = setInterval(function() {
          console.log("from inside clock")
          setSeconds(prev => prev + 1);
        }, 1000);
        // cleanup functions

        return function() {
          clearInterval(clock)
        }

    }, []);
```

with cleanup code will stop logging

Children

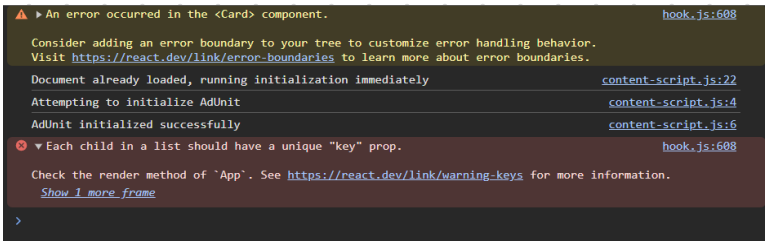
```
function App() {

    return <div style={{display: "flex"}}>
      <Card innerContent={<div style={{color:
      "green"}}>
        What do you want to post <br/> <br/>
        <input type={"text"} />
      </div>} />
      <Card innerContent={"hi there"} />
    </div>
}

function Card({ innerContent }) {
  return <div style={{background: "red",
  borderRadius: 10, color: "white", padding: 10,
  margin: 10}}>
    {innerContent}
  </div>
}
```

creating a wrapper class

```
function App() {

    return <div style={{display: "flex"}}>
      <Card>
        <div style={{color: "green"}}>
          What do you want to post <br/> <br/>
          <input type={"text"} />
        </div>
      </Card>
      <Card>
        hi there
      </Card>
    </div>
}

function Card({ children }) {
  return <div style={{background: "red", borderRadius: 10, color: "white", padding: 10, margin: 10}}>
    {children}
  </div>
}


export default App
```

(looks clean)

Lists and Key



without a Key there is a error

Key acts like a identity

Pass Key unique

Inline Styles

```
import React from 'react';

const App = () => {
    return (
        <div>
            <MyComponent />
        </div>
    );
};

const componentStyles = { backgroundColor: 'red',
color: 'white' }
function MyComponent() {
  return (
    <div style={componentStyles}>
      Hello, World!
    </div>
  );
}

export default App;
```

Object → <div style={{ backgroundColor: 'red', ... }}>

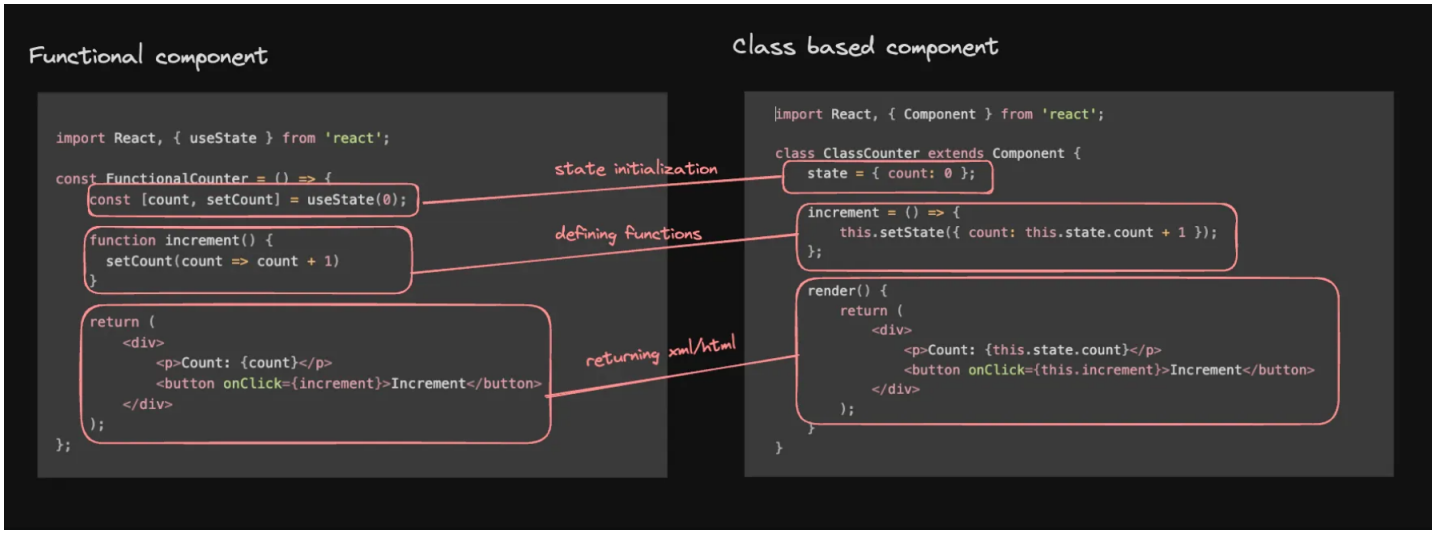
this is why two brackets

background-color → backgroundColor

html → JS

Class Based vs functional Components

↳ Interview qus



This i used till now

Same like C++ useStateX hooksX

Lifecycle events → Introduced in class based component

```jsx
import React, { useState, useEffect } from 'react';

function MyComponent() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    console.log('Component mounted or count updated');

  }, [count]); // Runs on mount and when count changes

  useEffect(() => {
        console.log('Component mounted');
    return () => {
      console.log('Component will unmount');
    };
  }, [])

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

} lifecycle event mounts when () runs and unmounts when () removed.
↳ runs only when code mounts.

```jsx
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  componentDidMount() {
    console.log('Component mounted');
  }

  componentDidUpdate(prevProps, prevState) {
    console.log('Component updated');
  }

  componentWillUnmount() {
    console.log('Component will unmount');
  }

  render() {
    return (
      <div>
        <p>Count: {this.state.count}</p>
        <button onClick={() => this.setState({ count: this.state.coun
          Increment
        </button>
      </div>
    );
  }
}
```

→ Class based component

Error Boundary

↳ used when one component of a page stops working but other things works perfectly fine.

↳ like in yt studio → stats are not visible
↳ shows refresh please

↳ use to bound an error
↳ does not work in functional component need to use class based component
↳ use as a black box
↳ useful in mobile App

```jsx
import React from 'react';
const App = () =>{

    return (
        <div>
            <ErrorBoundary>
                <Card1 />
            </ErrorBoundary>
            <Card2 />
        </div>
    )
};
function Card1() {
    throw new Error("Error while rendering");
    return <div style = {{background: "red", borderRadius: 20, padding: 20}}>
        hi there
    </div>
}
function Card2() {
    return <div style = {{background: "red", borderRadius: 20, padding:20 ,
    margin:20}}>
        hemlo
    </div>
}
```

→ Any error website won't crash

take as a black box always useful

```jsx
class ErrorBoundary extends React.Component {
    constructor(props) {
        super(props);
        this.state = { hasError: false };
    }

    static getDerivedStateFromError(error) {
        return { hasError: true };
    }

    componentDidCatch(error, info) {
        console.error("Error caught:", error, info);
    }

    render() {
        if (this.state.hasError) {
            return <div style = {{background: "red", borderRadius: 20, padding:
            20}}>
                Something went wroing
            </div>
        }

        return this.props.children;
    }
}

export default App;
```

Something went wroing

hemlo

→ due to the error

Fragments

→ there must be one parent div

```jsx
function App() {
    return (
        <div>
            <div>hui there</div>
            <div> hello</div>
        </div>
    )
};
```

→

return (
<>
 \n \n
 \n \n
</>
)

fragments

Single Page Application

↳ web application to load single page
↳ dynamically update the page as interacts
↳ for smoother exp compared to traditional multi page applications
↳ MPA require a full page reload and this don't

eg → Allen Website (SPA)
↳ Google (MPA)

dom → Document Object Model

Routing

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import './App.css'
function App() {
    return<div>
        <a href ="/">Allen</a>
        |
        <a href ="/neet/online-coaching-class-11">Class 11</a>
        |
        <a href ="/neet/online-coaching-class-12">Class 12</a>
        <BrowserRouter>
            <Routes>
                <Route path = "/neet/online-coaching-class-11" element =
                {<Class11Program />} />
                <Route path = "/neet/online-coaching-class-12" element =
                {<Class12Program />} />
                <Route path = "/" element = {<Landing />} />
            </Routes>
        </BrowserRouter>
    </div>
}
function Landing(){
    return <div>
        Welcome to allen
    </div>
}
function Class11Program(){
    return <div>
        Neet 11
    </div>
}
function Class12Program(){
    return <div>
        Neet 12
    </div>
}
```

Bad way of doing routing because it loads the html whole file again
Becomes a MPA

a tag

To fix that we use
↳ Links
↳ Navigate

```
import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
import './App.css'
function App() {
    return<div>
        <BrowserRouter>
            <Link to ="/">Allen</Link>
            |
            <Link to ="/neet/online-coaching-class-11">Class 11</Link>
            |
            <Link to ="/neet/online-coaching-class-12">Class 12</Link>
            <Routes>
                <Route path = "/neet/online-coaching-class-11" element =
                {<Class11Program />} />
                <Route path = "/neet/online-coaching-class-12" element =
                {<Class12Program />} />
                <Route path = "/" element = {<Landing />} />
            </Routes>
        </BrowserRouter>
    </div>
}
function Landing(){
    return <div>
        Welcome to allen
    </div>
}
function Class11Program(){
    return <div>
        Neet 11
```

This does not reload whole page just brings the changed part

Link tag

```jsx
function Class12Program() {
  const navigate = useNavigate();

  function redirectUser() {
    navigate("/")
  }

  return <div>
      NEET programs for Class 12th
      <button onClick={redirectUser}>Go back to landing page</button>
  </div>
}
```

move from Class12 route to / route using useNavigate

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

Other way for routes

Creates a array with a for loop

```jsx
const router = [{
  path: "/neet/online-coaching-class-11",
  element:
}, {
  path: "",
  element
}]

return <div>
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<Layout />}>
        <Route path="/neet/online-coaching-class-11" element={<Class11Program />} />
        <Route path="/neet/online-coaching-class-12" element={<Class12Program />} />
        <Route path="/neet/online-coaching-class-12" element={<Class12Program />} />
        <Route path="/" element={<Landing />} />
        <Route path="*" element={<ErrorPage />} />
      </Route>
    </Routes>
```

→ Same things different ways

For 404 Page

↳ Introduce one more route

<Route path="*" element={<ErrorPage />} />

```
function ErrorPage(){
    return <div>
        Sorry Page not found
    </div>
```

↳ will show error message for all those routes which are not mentioned.

Layouts

```jsx
import { BrowserRouter, Routes, Route, Link, Outlet } from "react-router-dom";
import './App.css'
function App() {
    return<div>
        <BrowserRouter>
            <Routes>
                <Route path = "/" element = {<Layout />}>
                    <Route path = "/neet/online-coaching-class-11" element =
                    {<Class11Program />} />
                    <Route path = "/neet/online-coaching-class-12" element =
                    {<Class12Program />} />
                    <Route path = "/" element = {<Landing />} />
                    <Route path = "*" element = {<ErrorPage />} />
                </Route>
            </Routes>
            Footer | Contact us
        </BrowserRouter>
    </div>
}
function Layout(){
    return <div style = {{height: "100vh"}}>
        <Header />
        <div style = {{height:"90vh", background:"red"}}>
            <Outlet />
        </div>
        footer
    </div>
}
```

Wrap all routes inside

creating a layout

Header

shown content

Footer

needed

The content

```jsx
function Header(){
    return <div>
        <Link to ="/">Allen</Link>
        |
        <Link to ="/neet/online-coaching-class-11">Class 11</Link>
        |
        <Link to ="/neet/online-coaching-class-12">Class 12</Link>
    </div>
}
function ErrorPage() {
    return <div>
        Sorry page not found
    </div>
}

function Landing(){
    return <div>
        Welcome to allen
    </div>
}
function Class11Program(){
    return <div>
        Neet 11
    </div>
}
```