**12 JAN 24**

1. React is a JavaScript library for building user interfacescreated by Facebook..
2. React is used to build single-page applications.
3. React allows us to create reusable UI components.
4. React Work by creating a VIRTUAL DOM in memory.

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

1. React only changes what needs to be changed!

React finds out what changes have been made, and changes only what needs to be changed.

1. Latest version of React is 18 as per 2024
2. To use React in production, you need npm which is included with [Node.js](https://nodejs.org/).

But in order to use React in production, you need npm and [Node.js](https://nodejs.org/) installed.

1. *Best Part of react is that it has component and that’s why it is reusable.*
2. If you have npx and Node.js installed, you can create a React application by using create-react-app.

*npx create-react-app my-react-app*

*cd my-react-app*

*npm start*

1. A new browser window will pop up with your newly created React App! If not, open your browser and type localhost:3000 in the address bar.
2. ***Single page application/reusibility/Virtual DOM***
3. *JSX , Props, Styling, Destructing, ES6, Declarative Programming, Hooks, COmponents, useState, import, Export, Map, Filter, virtual DOM, COntainer, Babel, Spread Operator*
4. *Codesandbox - A cloud development environment for react*
5. *Babel is a js compiler used to convert ECMAScript 2015+ code into backwards-compatible JavaScript code that can be run by older JavaScript engines.*

*I.e. convert next generation js to browser compatible* js

1. *Now in ES6 feature we can write import in place of require*

*var React = require("react");*

*var ReactDOM = require("react-dom");*

1. *Interpolation*

*Var x= “Aman”*

*<h1>My name is{x}</h1>*

1. *Expression vs statement -*

Statements are typically instructions that perform an action or control the flow of execution. They don't necessarily return a value.

*/ Variable declaration statement*

*let x;*

*// Assignment statement*

*x = 10;*

*// Conditional statement*

*if (x > 5) {*

*// code block*

*}*

*// Loop statement*

*for (let i = 0; i < 5; i++) {*

*// code block*

*}*

An expression, on the other hand, is a syntactic unit that produces a value

*/ Arithmetic expression*

*let result = 3 + 5;*

*// Function call expression*

*let length = calculateLength(4, 5);*

*// Conditional expression (ternary operator)*

*let isEven = (x % 2 === 0) ? 'Even' : 'Odd';*

1. *Template literal- hello my name is ‘${fname} ${lname}’*
2. *Component is just time functions in javascript*

**SPA vs MPA**

1. SEO is not Good in SPA as compare to MPA
2. In MPA there is POST request and response HTML while in SPA there is AJAX request and JSON response Eg. Netflix, youtube, facebook, whatsapp
3. React is a declarative Language means only tell what you want while javascript is a imperative Language means u have toi describe each and every steps i.e. What u want to do and how u want to do

**13 JAN 24**

1. Development vs Production - Development CDN are those which are used at the time of development because it will show and highlight all the errors so that developers can resolve it

While Production CDN are those which are used when we deploy and make it available for the users then we did not need to shows the errors as it will directly crash.

1. CDN - content delivery network
2. React Element -

Const x = React.CreateElement(type,{Element Name}, text)

Const x = React.CreateElement(“H1”,{ClassName:”head”},”Hello React”)

1. Difference between a react element and js element is that js element return a DOM node like H2 and classname and all but react return a object contains props which contain class and id and also content as a chindrenName
2. So, if u want to add react element directly into the DOM

ReactDOM.createRoot(document.getElementById(“root”)).render(reactHeading);

SO, this will make a virtual DOM

**16 JAN 24**

**Virtual DOM**

1. There is a Document object model for each node of the HTML element which is a tree like structure and connect the elements
2. Virtual DOM is a programming concept where a virtual representation of UI is kept in memory and it sync with the real DOM
3. There are two virtual DOM pre Updated and Updated So, react maintain two virtual DOM

**JSX and Babel**

1. JSX is a javascript XML i.e. Javascript and HTML but not compatible for the broswer
2. Babel is a compiler that convert JSX to javascript
3. So, to use Babel either install it or include its script CDN in the head
4. And after that for that particular js file u need to mention the type as text/babel

**17 JAN 24**

**React Fragment**

1. We can not do like this

Const x = 1 2

If u have to assign to variable like this u need to create a parent element and place these in that parent as a child

Const x =

<div>

1

2

</div>

Or use <React.Fragment> at place of div(parent element)

Or just use <> and </>

**React Component**

1. There are two ways of creating a react component

Functional Component - just like function in js

Class Based Component - just like class in js

1. In component function the function name should be start with a capital letter
2. This will tell the browser that it is a react component and not a native component
3. If u call that component like name() then it will consider as a function and not a component so call it like <name /> this
4. Arrow function implicitly return we do not need to write return unlike in normal functions

**Skiip To Hooks**

**React Hooks**

1. Hooks allow us to use the stateful logic without disturbing the hierarchy of the app

Create all related logic together(Separation of concern)

1. So, Hooks are the functions that let you hook into React state and lifecycle features from functional components.
2. Hooks can not be used in class based component

**React UseState**

**React Routing**

1. Moving from one page to other in a single page application
2. There are two type of routing Server side and client side.

Server side is mostly for MPA and Client side is for SPA

1. React-Router-DOM

**18 JAN 24**

1. *contentEditable is used to change content of that element*
2. *style={{property:”value”}}*
3. *const x={*

*border:”1px solid black”*

*}*

*x.border = “2px”;*

*style={x}*

1. Save it with a name.jsx file for component and write the functional component code

now we can use it in all ovee the apps by tag <name />

export default name;

Now import that component in the file where u have to use as

import name form :”./name”;

**19 JAN 24**

1. In default export we can change the name

export default pi;

export {dpi,tpi}

Pi name during import can be anything but dpi and tpi name should be same

Also use pi as {pi} while dpi and tpi as {dpi()} because the are function

1. Import \* as pi from “./location”

<ul>

<li>{pi.default}</li>

<li>{pi.dpi()}/li>

<li>{pi.tpi()}</li>

</ul>

**22 JAN 24**

1. Difference between functional and class component.
2. Hooks
3. Fundamental of reacts
4. Props
5. How to call from onr onr component to other without using props,hooks
6. useState is React Hook that allows you to add state to a functional component. It returns an array with two values: the current state and a function to update it. The Hook takes an initial state value as an argument and returns an updated state value whenever the setter function is called

**30 JAN 24**

**Embebde JSX**

1. Function App(){

Var x=”word”

return(

<>

<p>Hello {x}</p>

</>

)

}

1. Function x(){

return value

}

return(

<>

<h1>{x}</h1>

<p>

</>

)

**Rendering Arrays and objects**

1. Arrays are ok but Object can not be access as a react child like {obj} this
2. But {obj.name} this will give value of name property but can not whole object
3. For loop does not work in react because it does not return anything
4. That’s why we use map
5. {arr.map(num)=> <h2>{num}</h2>)}

**Assigning unique key**

1. React expect that in a array of string each element is assign with a unique key
2. So in array we can get unique key by using the array index
3. Function App(){

Let cars ={“A”, “B”, “c”};

return(

<>

<ul>

{cars.map((car,index)=>(

<li key={index}>{car}<li>

))}

</ul>

</>

)

}

1. <thead> and <tbody> tags are also used when u r creating table else it will show error that tr can not be the child of table

**Conditional rendering**

1. Using if
2. Using ternary operator - if u need a return inside the return then use ternary operator

**AND OR operator**

1. In js every thing except 0, null, “ ”,undefined and false is true
2. && return us last truthy value

Eg. true && hello = hello and hello && true = true

1. And also it returns first falsy value
2. OR returns last falsy and first truthy

**How to Use Events in jsx or react**

1. When app is render it first call the function regardless of the event is happened or not and thats why even if you not click the function will be render which you where expecting to render after the click the button
2. So in curly braces do not call the function linke this {nameodfunction()}

Instead of this remove parentheses i.e. only give the reference

**Virtual DOM under Hooks**

<!DOCTYPE html>

<html lang="en">

<head>

<script

crossorigin

src="https://unpkg.com/react@18/umd/react.development.js"

></script>

<script

crossorigin

src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"

></script>

<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

<title>React App</title>

</head>

<body>

<div id="root"></div>

<script type="text/babel">

const setShow = () => {};

const App = () => {

const [showParagraph, setShowParagraph] = React.useState(true);

const toggleParagraph = () => {

setShowParagraph((prevShowParagraph) => !prevShowParagraph);

};

return(

<>

<h1>Coding Ninjas</h1>

<button onClick={toggleParagraph}>

{showParagraph ? 'Hide Paragraph' : 'Show Paragraph'}

</button>

{showParagraph && (

<p>

This is the paragraph that will be toggled. Click the button

above to hide or show this paragraph.

</p>

)}

</>

);

};

const rootElement = ReactDOM.createRoot(document.getElementById("root"));

rootElement.render(<App />);

</script>

</body>

</html>

**5 FEB 24**

**Event handling**

1. During event handling you should do not follow function with parameter instead of this follow function reference or inline function

import React, { useState } from "react";

// var score=0;

var wicket=0;

function Score(){

let [score,setScore] = useState(0);

function addOne(){

setScore(()=> score+=1)

}

function addnum(num){

setScore(()=> score+=num)

}

return(

<div>

<h1>Score Keeper</h1>

<h1>Score:{score}/{wicket}</h1>

<button onClick={addOne}>1</button>

<button onClick={()=>addnum(2)}>2</button>

<button onClick={()=>addnum(3)}>3</button>

<button onClick={()=>addnum(4)}>4</button>

<button onClick={()=>addnum(5)}>5</button>

<button onClick={()=>addnum(6)}>6</button>

<button>Submit</button>

</div>

)

};

export default Score;

1. Mini project

import React, { useState } from "react";

// var score=0;

function Score(){

let [score,setScore] = useState(0);

let [wicket,setWicket] = useState(0);

function addOne(){

if(wicket<10)

setScore(()=> score+=1)

}

function addnum(num){

if(wicket<10)

setScore(()=> score+=num)

}

function getwicket(){

if(wicket<10)

setWicket(()=> wicket+=1)

}

return(

<div>

<h1>Score Keeper</h1>

<h1>Score:{score}/{wicket}</h1>

<button onClick={addOne}>1</button>

<button onClick={()=>addnum(2)}>2</button>

<button onClick={()=>addnum(3)}>3</button>

<button onClick={()=>addnum(4)}>4</button>

<button onClick={()=>addnum(5)}>5</button>

<button onClick={()=>addnum(6)}>6</button>

<button onClick={()=>getwicket()}>Wicket</button>

</div>

)

};

export default Score;



function ScoreButtons(props) {

const {getwicket,addnum} = props

return (

<div>

<button onClick={() =>{addnum(1)} }>1</button>

<button onClick={() =>{addnum(2)} }>2</button>

<button onClick={() =>{addnum(3)} }>3</button>

<button onClick={() =>{addnum(4)} }>4</button>

<button onClick={() =>{addnum(5)} }>5</button>

<button onClick={() =>{addnum(6)} }>6</button>

<button onClick={() => getwicket()}>Wicket</button>

</div>

)

}

export default ScoreButtons;

import ScoreButtons from "./ScoreButtons";

import React, { useState } from "react";

function Score() {

let [score, setScore] = useState(0);

let [wicket, setWicket] = useState(0);

function addnum(num) {

if (wicket < 10)

setScore(() => score+= num)

}

function getwicket() {

if (wicket < 10)

setWicket(() => wicket+= 1)

}

return (

<div>

<h1>Score Keeper</h1>

<h1>Score:{score}/{wicket}</h1>

{/\* <ScoreButtons /> \*/}

<ScoreButtons addnum={addnum} getwicket={getwicket}/>

</div>

)

};

export default Score;

1. **Storing the project in a array**
2. Index and view
3. **Map** - Create a new array by doing something with each item in an array.

**Filter** - Create a new array by keeping the items that return true.

**Reduce** - Accumulate a value by doing something to each item in an array.

**Find** - find the first item that matches from an array.

**FindeIndex** - find the index of the first item that matches.

1. map returns a new array with the results of the function, while forEach does not return anything and only modifies the original array.

// var numbers = [3, 56, 2, 48, 5];

// const newNumbers = numbers.map(x => x \* x);

//Map -Create a new array by doing something with each item in an array.

// const newNumbers = numbers.map( x => x \* 2);

////Filter - Create a new array by keeping the items that return true.

// const newNumbers = numbers.filter(num => num < 10);

//Reduce - Accumulate a value by doing something to each item in an array.

//const newNumber = numbers.reduce((accumulator,currentNumber) => accumulator + currentNumber);

//Find - find the first item that matches from an array.

// const newNumber = numbers.find(num => num > 10);

//FindIndex - find the index of the first item that matches.

// const newNumber = numbers.findIndex(num => num > 10);

1. mapfilterreduce

<https://codesandbox.io/p/sandbox/mapfilterreduce-completed-3sm6u?file=%2Fsrc%2Findex.js%3A50%2C22>

1. .conditional-rendering-completed

<https://codesandbox.io/p/sandbox/conditional-rendering-completed-ovu1v?file=%2Fsrc%2Fcomponents%2FApp.jsx%3A3%2C1>

1. Usestate Hook

<https://codesandbox.io/s/usestate-hook-completed-ylxqj>

1. Destructuring array, object

<https://codesandbox.io/p/sandbox/es6-destructuring-completed-zvjn9?file=%2Fsrc%2Findex.js>

1. Event handling

<https://codesandbox.io/p/sandbox/event-handling-in-react-completed-61rti?file=%2Fsrc%2Findex.js>

1. React- form

<https://codesandbox.io/p/sandbox/react-forms-23oen?file=%2Fsrc%2Findex.js>

import React, { useState } from "react";

function App() {

const [name, setName] = useState("");

const [headingText, setHeading] = useState("");

function handleChange(event) {

setName(event.target.value);

}

function handleClick(event) {

setHeading(name);

event.preventDefalut();

}

return (

<div className="container">

<h1>Hello {name}</h1>

<input

onChange={handleChange}

type="text"

placeholder="What's your name?"

value={name}

/>

<button type="submit">Submit</button>

</div>

);

}

export default App;

1. Changing complex component

<https://codesandbox.io/p/sandbox/changing-complex-state-completed-3hyn7?file=%2Fsrc%2Findex.js>

1. ES6 spread operator

<https://codesandbox.io/p/sandbox/es6-spread-operator-completed-3w3pp?file=%2Fsrc%2Findex.js>