React

React is a JS library created by facebook. React is a tool for ~~to~~ building UI components. It creates 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. React only changes what needs to be changed.

Need of React

1) Concept of Virtual DOM
2) Large Community
3) Industry standard
4) make easily websites

Setup React JS

1) Requirements (npm, node js)
2) Open code editor (Vs code) and Type in its terminal

```
npx create-react-app name-of-app↵
```

folder structure of React

1) README.md (for read instructions)
2) package.json (contain information about project and also about dependencies that installed in application)
3) public folder (contain a bunch of static files and index.html file)
4) src folder (contain files we work with)

Start an application

to run react app, go to terminal and type 'npm start↵'

build structure of index.js "src > ~~App~~ index.js"

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<React.StrictMode> <App /> </React.StrictMode>);
```

```
import "./App.css";
function App(){
    return(<>  </>;)
} export difault App;
```
← App.js

Note:- All the code of web page is written in App.js file (components can be imported)

JSX

1) JavaScript function that return HTML
2) Syntax represented by react, allows us to rendering logic with ui
3) Reusable components and variables.

```
function App(){
    const name =<h1>Bachan</h1>;
    const age = <h2>21<h2>;
    ~~return(~~
    Const user = (<div>{name}{age}</div>);
    return (<> {usr} </>)
} export default App;
```
← JSX example

Component

It is a JS function that return some UI or JSX (always).

Note:- component name start with capital letter

JS function →
```
Const getName =() =>{
        return "Bachan";
};
```

```
Const getNameComponent =() =>{
    return <h1>Bachan</h1>;
};
```
↑ JSX Component

```
Const Usr = () => { return ( <>
        <h1> Bachan </h1>
        <h2> 21 </h2>
};
function App(){ return ( <>
        <Usr /> </>;
}
```
← JSX Component

Props

1) It is a JS object that exist, exsist of arguments of some components.
2) It is the data that we share through components.
3) In the object we represent all the data that might want to pass between a component.
4) We can pass any kind of data.

← Props Example

```
function App(){
    return ( <>
        <User name="Bachan" age={21} />
        <usr name="Sugam" age={21} />
    </>);
}
Const usr =(props) => {
    return (<>
        <h1>{props.name} </h1>
        <h1> {props.age} </h1>
    </>);
} export default App;
```

Implimenting Css

1) import using "import './App.css';" or
2) import using "import styles from './App.module.css';" for App.module.css in we use cade css

```
<div className={styles.App}> Bachan </div>
```

Conditional Rendering

```
<> {age ≥ 18 ? <h1> over age </h1> : <h2> Under age </h2> }</>
```

```
Const isGreen = false;
return(<>
    <h1 style={{ color: isGreen ? "greed": "red"; }}> text </h1>
    { isGreen && <button> submit </button> }
```

Lists

plays important role while working with complex projects.

```
function App(){
    const names =["bachan","sugam","Nitil","Sachan"];
    return(<>
        {names.map((name,key)=>{
            return <h1 key={key}>{name}</h1>;
        })
    }</>);
}export default App;
```

create file under src folder

App.js

```
import "./App.css";
import {user} from "./user";
function App(){
    const users=[
        {name:"Bachan",age:21},
        {name:"sugam",age:21},
    ];
return(<>
    {user.map((user,key)=>{
        return(<User name={user.name}
        age={user.age} /> })
    }</>
);}export default App;
```

src\user.js

```
export const User =(props)=>{
 return(
  <>
  {props.name}
  {props.age}
 </>
);
};
```

States Hook

useState declares a state variable that we can update directly It is just a variable that holds the data.

increment on button onClick →

```
import {useState} from "react";
function App(){
        const [age, setAge] = useState(0);
        const increaseAge =()=>{ setAge(age+1); };
return(<> {age}
        <button onClick={increaseAge}>Increment</button>
</>);}export default App;
```

```
import {useState} from "react";
function App(){
    const [inputVal, setInputVal] = useState("");
    const handleInput = (event) => {
        setInputVal(event.target.value);
    }; return(<>
        <input type="text" onChange={handleInput} />
        {inputVal} </> );
} export default App;
```

```
import {useState} from "react";
function App(){
    const [showText, setShowText] = useState(true);
    return(<>
        <button onClick={()=>{setShowText(!showText);}}>
        show/hide </button>
        {showText && <h1>JAI SRI RAM </h1> </>); }
export default App;
```

```
import {useState} from "react";
function App(){
    const [textColor, setTextColor] = useState("black");
    return(<>
        <button onClick={
            ()=>{ setTextColor(textColor === "black"?
                "red": "black"); }}> show color </button>
        <h1 style={{color: textColor}}>
                Text to change color </h1>
    </> );};
    export default App;
```

Lifecycle of Component

3 stages
- mounting (component appearing or starting existing)
- updating (component is changing ~~dupa~~depend upon event)
- unmounting (component stop appearing on screen)

App.js

```
import './App.css';
import {useState} from "react";
import {Text} from "./Text";
function App(){
  const [showText, setShowText] = useState(false);
  return (<>
  <button onClick={() => {
      setShowText(!showText);}}>
  Show Text </button>
{showText && </Text /> } </>
); } export default App;
```

Text.js

```
import React from "react";
import {useState} from "react";
export const Text =() => {
  const [text, setText] = useState("");
  return (<>
  <input onChange={(event)=>{
    setText(event.target.value);
  }} />
<h1>{text}</h1>
); };
```

useEffect Hook

Effect let a component connect to and asychronize with external system. This includes dealing with network, browser DOM, animation, widgets written using a different UI library. It connects a component to an external system

example of useEffect →

```
function ChatRoom({roomId}){
    useEffect(()=>{
      const connection = createConnection(roomId);
      connection.connect();
      return () => connection.disconnect();
  }, [roomId]);
```

1) used to control what happend depending on which stage of component lifecycle is.
2) we create action that triggered when component is mounting, updating and unmounting.

Start Activity (msg) once → `useEffect(() => { console.log("Mounted"); }, []);`

Anytime → `useEffect(() => { console.log("Mounted"); }, [text]);`

```
useEffect (() => { consol.log ("Mounted");
        return () => { consol.log ("Unmounted"); }; }, []);
```

↳ return when component stops

<React.StrictMode>

1) It is used to help us write better code
2) It makes some check while writing code and prevent to make mistake.

How to Fetch data using API

1) fetch() used to fetch data from API, and it returns a promise as JS object

```
fetch ("API Link"). then ((res) =>
        res.json(). then ((data) => { //content });
```

2) Axios library helps to fetch data from API

```
npm install axios
```

3) import that axios library using import axios from "axios";
4) to accuss data use code

```
Axios.get("API Link"). then ((res) => {
        console. log (res.data); }) };
```

5) use code to stop rendering data infinity time

```
useEffect ( () => {
    Axios.get("API Link"). then ((res) => {
    setCatFact(res.data.fact); }) }, []);
```

6) use function on button onClick and call that using

```
useEffect( () => { fetchCatFact(); }, [ ]);
<p> { catFact } </p>
```

Example of API using React:

```
import Axios from "axios";
import { useEffect, useState } from "react";
function App(){
        const [name, setName] = useState(" ");
        const [predictAge, setPredictAge] = useState(null);
        Const fetchdata = () => {
                Axios.get(" link... /?name = ${name} ").then(
                (res) => { setPredictAge(res.data); }); }; ...
<input placeholder = "name" onChange = {(event) => {
        setName (event.target.value); }} />
<button onClick = {fetchData}> Predict Age </button>
<h1> Name : {predictAge?.name} </h1>
<h3> {PredictAge?.age} </h3>
```

Note:- ? (if not null then return value) present in variable

react-router-dom Liberary

used to create routes in react app (widely used)

```
npm install react-router-dom
```

```
import { BrowserRouter as Router, Routes, Route } from
                "react-router-dom";
```

import Routes

Routes: It is way to only change a specific portion of a page.

function App () { return (<>

<Router>: to tell React/DOM, whatever is inside can use the react or DOM functionlity. used to define where in our app we want to access to DOM element.

<Routes>: is used to describe the individual <Route> inside it.

```
import {HOME} from "./pages/Home";
function App() { return ( <>
    <Router> <Navbar /> <Routes>
    <Route path="/" element={ <Home /> } />
    <Route path="/about" element={ <About /> }
    </Routes> </Router> </>); }; export default App;
```

App.js

```
import React from "react";
import {Link} from "react-router-dom";
const Navbar = () => { return (<>
    <Link to="/"> Home </Link>
    <Link to="/about"> About </Link>
    <Link to="/menu"> Menu </Link> </> }
export default Navbar;
```

Navbar.js

State Management

1) Solve the issue of prop drilling
2) prop drilling is concept, idea of passing a props to component and that prop can't use in component, it been a purpose to ~~join~~ just to pass another component.
3) State management helps to orgnise project.

useContext Hook

Context API allows us to put all of the states functions and variables at one place and then let all the component inside that place have access to it directly without having to use props.

```
import { useState, createContext } from "react";
export const AppContext = createContext();...
<> <AppContext.Provider value={{ username, setUsername }}>
... </AppContext.Provider> </>
```

```
import { useContext } from "react";
import { AppContext } from "../App";
export Home = () => {
    const {username} = useContext(AppContext);...
    <H1> {username} </H1>...
```

React Query

```
npm install @tanstack/react-query
```

```
import {QueryClient, QueryClientProvider} from "@tanstack/react-query";
function App () {
    const client = new QueryClient();
    return (<>
    <QueryClientProvider client={client}>...
    </QueryClientProvider>...
```

```
import {useQuery} from "@tanstack/react-query";
import Axios from "axios";
export const Home =() => {
    const {data} = useQuery(["cat"], () => { return
        Axios.get("Link API").then((res) => res.data); })...
        {data?.fact}
    const {data, isLoading} = ... });
    if (isLoading) {return <h1> Load </h1>}
    const {data, refetch} = ...
    <button onClick={refetch}> Update data </button>
    const {data : catData,} ... <p> {catData?.fact} </p>
```

Configuration of React Query (immutation)

Page switch, data will not update.

```
Const client = new QueryClient ({defaultOptions : { queries : {
        refetchOnWindowFocus : false, }, }, }); return
```

React Forms

Yup used to validate react-hook-form allow functionality to show errors and submit form etc.

```
npm install react-hook-form yup
npm install @hookform/resolvers
```

help us integration with yup and react-hook-form

```
import { useform } from "react-hook-form";
import * as yup from "yup";
import { yupResolver } from "@hookform/resolver/yup";
export const Form = () => {
    const schema = yup.object().shape({
    fullName : yup.string().required("Enter name"),
    email : yup.string().email().required("Enter email"),
    age : yup.number().positive().integer().min(18).required(),
    password : yup.string().min(4).max(20).required(),
    confirmPassword : yup.string().string().oneOf([yup.ref("
        password"), null], "Password not match").
        required(), });
const { register, handleSubmit, formState : { errors }, } =
    useForm ({ resolver : yupResolver(schema), });
const onSubmit = (data) => { console.log(data); } return (<>
<form onSubmit = {handleSubmit(onSubmit)}>
<input type = "text" {...register("fullName")} />
  <p> {errors.fullName?.message} </p>
<input type = "email" {...register("email")} />
  <p> {errors.email?.message} </p>
<input type = "number" {...register("age")} />
```

```
<p>{errors.age?.message}</p>
<input type="password" {...register("password")} />
<p>{errors.password?.message}</p>
<input type="password" {...register("confirmPassword")} />
<p>{errors.confirmPassword?.message}</p>
<input type="submit" /></form></> ); };
```

~~Con~~ Custom Hook

Simple useState Hook Example →

```
function App(){
  const [isVisible, setIsVisible] = useState(false);
  return(<>
  <button onClick={()=>setIsVisible((prev)=>!prev)}>
  {isVisible ? "Hide":"Show"}</button>
  {isVisible && <h1>Hidden Text</h1></>);}
```

1) Create custom file named with 'use' start
2) allways call in react component
3) Can't call inside function
4) Hook are used to handle logic (can't return TSX)

src > component > useToggle.js

useToggle.js (Custom Hook Design)

```
import {useState} from "react";
export const useToogle = (initialVal = false)=>{
  const [state, setState] = useState(initialVal);
  const toggle =()=>{ setState((prev)=>!prev); };
```

App.js

```
import "./App.css";
import {useToogle} from "./useToogle";
function App(){ <>
  <button onClick={toggle}>{isVisible ? "Hide":"Show"}
  </button>{isVisible && <h1>Text</h1>}</>);}
```

5) We can return object with {} from custom Hook.

TypeScript

install typeScript

npm install prop-types

import typeScript

import PropTypes from "prop-types";

create react app using typeScript

npx create-react-app . --template typeScript

we have to define type variable in typeScript

Interface : is a way to describe a shape.

```
import React from "react";
import "./App.css";
import {Person, Country} from "./components/Person";
function App(){ return (<>
  <Person name="Bachan" email="bachan@gmail.com"
  age={21} isMarried={true} friends={["ABC", "BCD", "DEF"]}
  country={Country.France} /> </> ); }
```

```
import {useState} from "react";
interface Props{ name: string; email: string; age: number;
  isMarried: boolean; friends: string[]; country: Country; }
export enum Country { France = "France", India = "India", }
export const Person = (props: Props) => { return(<>
  <h1> Name: {props.name}, Email: {props.email},
      Age: {props.age}, This Person {props.isMarried? "is":
    "isNot"} Married; friend are {props.friends.map(
    (friend: string) => { {friend} ))}
  Country: {props.country} </h1> </> ); };
```

Redux Toolkit (mange States)

npm install @reduxjs/toolkit react-redux

```
import {configureStore, createSlice} from "@reduxjs/toolkit";
const initialState = {value: {username: ""}};
const usrSlice = createSlice({name: "usr", initialState,
    reducers: {login: (state, action) => {state.value = action.payload;},
    logout: (state) => {state.value = initialState.value;},
},}); export const {login, logout} = usrSlice.actions;
export const store = configureStore({reducer: {
        usr: usrSlice.reducer,},});
```

```
import {Login} from "./pages/Login";
import {Provider} from "react-redux";
import {store} from "./store";
function App(){ return(<>
  <Provider store={store}> <Router> <Link...> <Routes> <Route...

import {useState} from "react";
import {login, logout} from "../store";
import {useDispatch, useSelector} from "react-redux";
export const Login = () =>{
    const [newVusname, setNewVusname] = useState("");
    const dispatch = useDispatch();
    const usname = useSelector((store: any) =>
                    state.value.usname);
    return(<> <h1>{usname}
    <input onChange={(e)=>{setNewVusname(e.target.value);}}/>
    <button onClick={()=>dispatch(login({username: newVusname
    }))}>Login</button>
    <button onClick={()=>dispatch(logout())}>
        Logout</button></h1></>); };
```

Login.jsx