# ReactJS

**ReactJS is a JavaScript library** used for building **reusable UI components**.

According React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by **Facebook**.

React is a tool for building UI components.

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 is a JavaScript library for building user interfaces.

React is used to build **single-page applications.**

React allows us to create **reusable** UI components.

# React.JS History

**React.JS was first used in 2011 for Facebook's News Feed feature.**

Initial Release to the Public (V0.3.0) was in July 2013.

Facebook Software Engineer, Jordan Walke, created it.

Current version of React.JS is V17.0.2 (August 2021).

React 18 was released in **March 2022**.

Current version of `create-react-app` is v4.0.3 (August 2021).

# React Advantages

- Uses **virtual DOM** which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.
- Can be used on client and server side as well=> as with other frameworks.
- Component and data patterns improve readability, which helps to maintain larger apps.

Why React

**Virtual Dom** in react so faster from real dom

Difference

Virtual dom work very smartly only & light weight. Also only affect one time on not again again for the whole element.

Reusable Web component

Maintained By Facebook

Hirable for fresher Students

==========

Like Example

Collection of webpages : https://www.makemytrip.com/

home.html

about.html

blog.html

contact.html

**SPA single page application**

index.html => all pages/**component**

==========

# Setting up a React Environment

Step :1 **node.js & npm** Install

To get an overview of what React is, you can write React code directly in HTML.

But in order to use React in production, you **need NPM and Node.js** installed.

If you have NPM and Node.js installed, you can create a React application by first installing the create-react-app.

npm stands for Node Package Manager. It's a library and registry for JavaScript software packages. npm also has command-line tools to help you install the different packages and manage their dependencies.

**https://nodejs.org/en/download/  download & install**

then check version

cmd : **node -v**

cmd: **npm -v**

**Step : 2  Download editors (vs code/ sublime)**

**Step : 3 Using node js install the create-node-app package**

**cmd: C  // only one time in pc**

# Setting up a React Environment

**If you have NPM and Node.js installed,** you can create a React application by first installing the create-react-app.

# Install React / Two Types

1) CDN file link

```
<script
src="https://unpkg.com/react@16/umd/react.production.min.js"></
script>
<script
src="https://unpkg.com/react-dom@16/umd/react-dom.production.mi
n.js"></script>
<script
src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></
script>
```

2) Instal by npx

Install create-react-app by running this command in your terminal:

```
C:\Users\          >npm -g install create-react-app
```

**cmd:  create-react-app --version**

Then you are able to create a React application, let's create one called `myfirstreact`.

Run this command to create a React application named `myfirstreact`:

```
C:\Users\Your Name>npx create-react-app myfirstreact
```

```
cmd : create-react-app myfirstreact
```

The create-react-app will set up everything you need to run a React application.

# Run the React Application

Now you are ready to run your first *real* React application!

Run this command to move to the `myfirstreact` directory:

```
C:\Users\Your Name>cd myfirstreact
```

Run this command to run the React application `myfirstreact`:

```
C:\Users\Your Name\myfirstreact>npm start

================================================================================
```

**React Folders Structure**

**node_modules: all react project framework code**

**public/Index.html : main first page load in react**

**Readme** - define command that what command work / so you can also delete

**Package.json** : main file for known about your project : like project name, all dependencies react version & script version , dom version & all

So if you transfer projects then you can manage env as per project.

**Package.lock.json : Define version schemas / Not any changes required**

**Src**: Main folder

**App.js is main component js file you can change html content & all**

So far so good, but how do I change the content?

Look in the `myfirstreact` directory, and you will find a `src` folder. Inside the `src` folder there is a file called `App.js`, open it and it will look like t his:

Try changing the HTML content and save the file.

**Also you can change content**

**Index.js : main inject js file first load**

If you want to follow the same steps on your computer, start by stripping down the `src` folder to only contain two files: `index.js` and `index.html`, in some builds you might find the index.html in the public folder instead, you should also remove any unnecessary lines of code inside the two files to make them look like the files in
the Show React tool below:
==========

**Now Version Change** : 18 release in april 2022, so we go for 17.0.2 as per course

Now go : package.json and change it and delete Node folder
"react": "^17.0.2",
"react-dom": "^17.0.2",

// again install npm means node module now all module set as per 17.0.2

CMD: npm install

Note: again all restart our project

Error: if you get any error : Import  React from 'react/client' in  index.js

then :
just remove client :  follow below

// in 17

```
import ReactDOM from 'react-dom';



// in 18
//import ReactDOM from 'react-dom/client';

// in 17
const rootNode = document.getElementById('root');

ReactDOM.render(<App />, rootNode);



/*
// in 18

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>

);
*/
```

==========

Download & use Extension in React

- **es7 + react**

JavaScript and React/Redux snippets in ES7+ with Babel plugin features for VS Code

Short code for program

imr    : import react

imrc : import react with component

**rfce : create function component**

**rce : create class component**

rcc :  create class component with export

rcep  : create class component with proptype

rcreduxp :

- **html-to-jsx** /  html  to  jsx

A basic Visual Studio Code plugin that converts html text to JSX string
You can select your html text and use the shortcut ctrl+alt+x

**There are many things to do**

- Add functionality to validate html code
- Add support to format the resulting string
- Add support to convert javascript strings to html code

==========

**How its work**

**App.js** : in this page an already created app component, so you can add any html content or code in this component. Which already display on the main page.

**Index.js :** this is main page means main component file this manage all pages & component // its call to <App>

**index.html  all index.js component load into index.html**
**pass id=”root”**

Now in this page call app component by ReactDOM.Rander() function & display in public/index.html by id

**Note : reactDom order to rander & rander function display data**

**// ReactDOM.render(para1 , para2)**
**// para 1 => what u want to show , kya dikhana H**
**// para 2 => where u want to show , kidr dikhana H**

**ReactDOM.render(<App />,document.getElementById('root'));**

**public/index.html :** Now app component display in this page by id

** **

**Node Module** : All virtual dom & other libraries folders
**Component : component means one time of any div or block sdo its just one js file we create, so we create multiple small component & create one website**

**Example :**

we creates components file like below
Header.jsx

```
import React from "react"   // load react in all component
function Header()
{
      return(
      <div>
            <h1>Header</h1>
      </div>
)}
```

```
export default Header;
```

Now if want to use all component in app.js

**App.js**

```
import React from "react"   // load react in all component
import Header from "./Header"  // load all component for use
import "./style.css"

function App()
{
        return(
        <div>
                < Header />
                </Main>
                < Footer />
        </div>
)}

export default App;
```

**index.js   main file which randerDom into index.html**

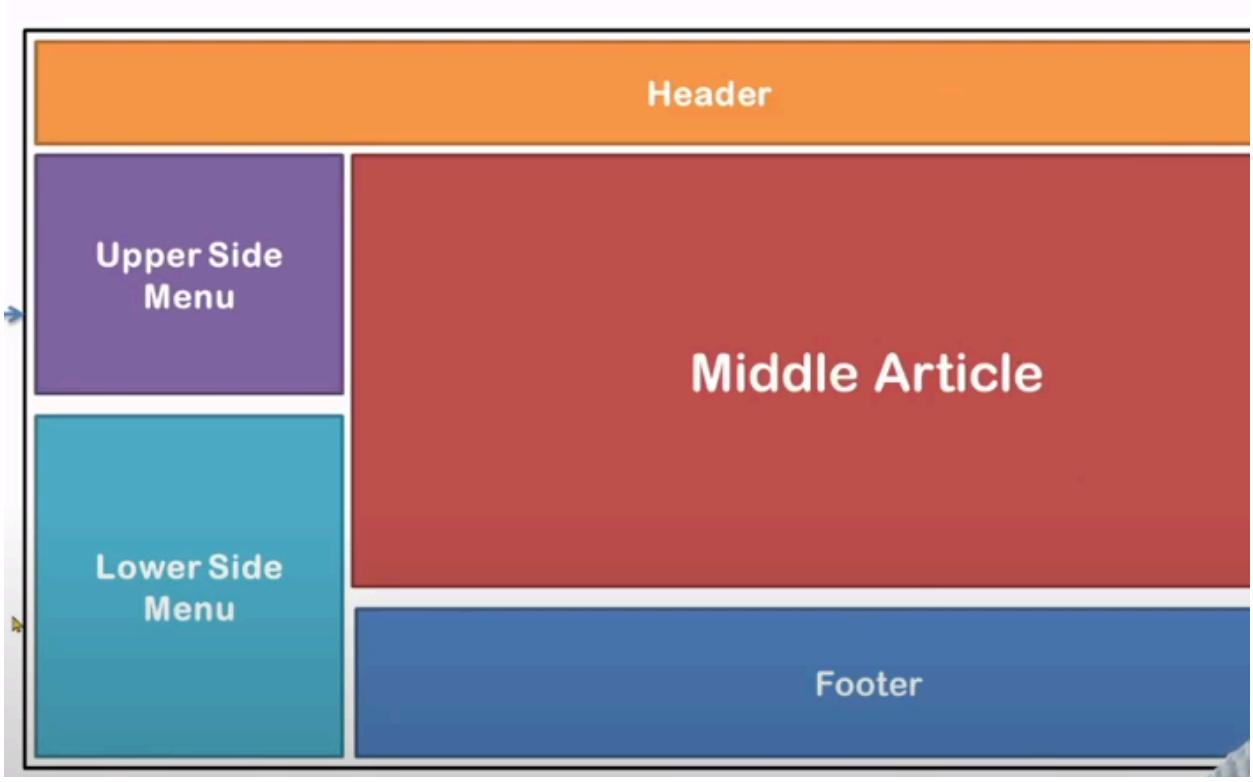
```
import React from "react"   // load react in all component
import ReactDom from "react-dom"
import Header from "./App";

ReactDOM.render(<App/>,document.getElementById('root'));
```

**public/index.html  // all component data show on below id**

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

# React Component?

- 
- 
- But Work in isolation and returns HTML via a render function.




Note: All custom component is child component of parent(index.js) component

React Components Types :

1] Functional Components:

In simple words, For Shirt functionality  Functional components are javascript functions.
By writing a javascript function, we can create a functional component in React Apps.
To make React app efficient, we use functional component only when we are sure that our component does not require to interact with any other components.

Functional components do not require data from other components.
Below shows an example of functional component in React:

1)

```
function Title()
{
  return <h1>I am Title</h1>;
}
export default Title
```

Again we can make by this method

2)

```
export const Man = () => {
        return <h1>I am Title</h1>;
}
```

```
/* now component ready now you can use anywhere
by
import Class_component from './Class_component';
then use direct as <Class_component/>
```

## 2] Class Components

Class component work as ES6 and manage states in class component

Used for big functionality , The class components are similar to the functional component but has some additional features that makes class component a little more complex than the functional components.

The functional components do not care about the other components in your app whereas the class components can work with each other.

We can pass data from one class component to other class component.
Below shows an example of class component in React:

We can use props and argument

```
import React from 'react';



import React, {Component } from 'react';

class Title extends React.Component  / Component
{

  constructor(props){}

   render(){
        return <h1>I am Title</h1>;
   }
}

export default Title


import Title from "./Component/Title";


/* now component ready now you can use anywhere by
import Class_component from './Class_component';
than use direct as <Class_component/>
*/
```

==========

also add props by constructor in class above rander()

```
constructor(props){
        super(props)
}
```

==========

What is JSX?

JSX stands for JavaScript XML.

JSX allows us to write HTML in React.

JSX makes it easier to write and add HTML in React.

Coding JSX

JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement()  and/or appendChild() methods.

JSX converts HTML tags into react elements.

Expressions in JSX

With JSX you can write expressions inside curly braces { }.

Example of declaration of javascript element & how to use in html direct

{print}

let data="Hi i am jsx";

{data}

const myelement = <h1>I Love JSX!</h1>;   {myelement }

const myelement = <h1>React is {5 + 5} times better with JSX</h1>;

Inserting a Large Block of HTML

const myelement = (

```
  <ul>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Cherries</li>
  </ul>
);
```

One Top Level Element

The HTML code must be wrapped in ONE top level element.

```
const myelement = (
  <div>
    <h1>I am a Header.</h1>
    <h1>I am a Header too.</h1>
  </div>
);
```

Elements Must be Closed

JSX follows XML rules, and therefore HTML elements must be properly closed.

```
const myelement = <input type="text" />;
```

==========

If WE GO WITHOUT JSX THEN WE HAVE TO DO BELOVE CODE

```
return React.createElement(
        "div",{ class: "header"},
         React.createElement("h1",null, "Welcome Raj")
)
```

==========

**Style IN JSX in component**

- By external

App.css => .header {color:red}

```
import   "./App.css";
<h1 className="header">Hello</h1>
```

- By Inline

```
render(){
const mystyle={fontSize:"20px”, color:"red" }
<h1 style={ mystyle }> Hello </h1>
or
<h1 style=”color:red”> Hello </h1>
<h1 style={{color:"red"}}> Hello </h1>
}
```

==========

# CSS Modules

Another way of adding styles to your application is to use CSS Modules.

CSS Modules are convenient for components that are placed in separate files.

The CSS inside a module is available only for the component that imported it, and you do not have to worry about name conflicts.

Create the CSS module with the `.module.css` extension, example: `my-style.module.css`.

## my-style.module.css:

```
.bigblue {
  color: DodgerBlue;
  padding: 40px;
  font-family: Sans-Serif;
  text-align: center;
}
```

Import the stylesheet in your component:

```
import styles from './my-style.module.css';
import styles1 from './my-style1.module.css';

const Car = () => {
  return <h1 className={styles.bigblue}>Hello Car!</h1>;
          or
 return <h1 className={styles1.bigblue}>Hello Car!</h1>;



}
```

==========

# Styling React Using Sass

## What is Sass?

- Sass stands for **Syntactically Awesome Stylesheet**
- Sass is an extension to CSS
- Sass is a CSS pre-processorSass is completely compatible with all
- versions of CSS
- Sass reduces repetition of CSS and therefore saves time
- Sass was designed by Hampton Catlin and developed by Natalie Weizenbaum in 2006
- Sass is free to download and use

---

## Why Use Sass?

Stylesheets are getting larger, more complex, and harder to maintain. This is where a CSS pre-processor can help.

Sass lets you use features that do not exist in CSS, like variables, nested rules, mixins, imports, inheritance, built-in functions, and other stuff.

If you use the `create-react-app` in your project, you can easily install and use Sass in your React projects.

Install Sass by running this command in your terminal:

```
>npm i sass
```

## Create a Sass file

Create a Sass file the same way as you create CSS files, but Sass files have the file extension `.scss`

In Sass files you can use variables and other Sass functions:

## my-sass.scss:

Create a variable to define the color of the text:

```
$myColor: red;

h1 {
  color: $myColor;
}
```

```
/* Define standard variables and values for website */
$bgcolor: lightblue;
$textcolor: darkblue;
$fontsize: 18px;

/* Use the variables */
body {
  background-color: $bgcolor;
  color: $textcolor;
  font-size: $fontsize;
}
```

Import the Sass file the same way as you imported a CSS file:

```
import './my-sass.scss';
```

# Sass Nested Rules

Sass lets you nest CSS selectors in the same way as HTM

In css

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
```

```
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}

in sass nested
nav {
  ul {
      margin: 0;
      padding: 0;
      list-style: none;
  }
  li {
      display: inline-block;
  }
  a {
      display: block;
      padding: 6px 12px;
      text-decoration: none;
  }
}
```

# Sass Nested Properties

```
in CSS

font-family: Helvetica, sans-serif;
font-size: 18px;
font-weight: bold;

text-align: center;
text-transform: lowercase;
text-overflow: hidden;

in sass

font: {
  family: Helvetica, sans-serif;
```

```
  size: 18px;
  weight: bold;
}

text: {
  align: center;
  transform: lowercase;
  overflow: hidden;
}
```

# Sass @extend and Inheritance

```
.button-basic  {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.button1  {
  @extend .button-basic;
  background-color: red;
}

.button2  {
  @extend .button-basic;
  background-color: yellow;
}
```

==========

Bootstrap 5

Link Load in index.html

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>

==========

# React Bootstrap

https://react-bootstrap.github.io/getting-started/introduction

The best way to consume React-Bootstrap is via the npm package which you can install with npm (or yarn if you prefer).

```
npm install react-bootstrap bootstrap
```

## Importing Components

You should import individual components like: react-bootstrap/Button rather than the entire library. Doing so pulls in only the specific components that you use, which can significantly reduce the amount of code you end up sending to the client.

```
import Button from 'react-bootstrap/Button';

// or less ideally

import { Button } from 'react-bootstrap';
```

```
import {

    Button,Table, Alert, Accordion, Badge, Breadcrumb, ButtonGroup, Card,
Carousel

    , Dropdown, Figure, ListGroup, Modal, Navbar, Nav, Container, NavDropdown,

    Form, FormControl, Row, Col

} from 'react-bootstrap';
```

<Button variant="primary">Primary</Button>

<Button variant="secondary">Secondary</Button>

<Button variant="success">Success</Button>

==========

# Material Bootstrap / MUI

# Move faster with intuitive React UI tools

MUI offers a comprehensive suite of UI tools to help you ship new features faster. Start with Material UI, our fully-loaded component library, or bring your own design system to our production-ready components.

Install Material UI, the world's most popular React UI framework.

## Default installation

```
npm install @mui/material @emotion/react @emotion/styled
```

`@mui/icons-material`

`@mui/x-data-grid`

Please note that react >= 17.0.0 and react-dom >= 17.0.0 are peer dependencies.

Material UI is using emotion as a styling engine by default. If you want to use `styled-components` instead, run:

==========

## With styled-components

```
npm install @mui/material @mui/styled-engine-sc styled-components
```

Use Exa:

```
import Button from '@mui/material/Button'
```

==========

# Styled Component

## https://styled-components.com/

https://styled-components.com/docs/basics

styled-components is the result of wondering how we could enhance CSS for styling React component systems. By focusing on a single use case we managed to optimize the experience for developers as well as the output for end users.

Apart from the improved experience for developers, styled-components provides:

Automatic critical CSS

No class name bugs:

Easier deletion of CSS:

Simple dynamic styling:

Painless maintenance:

## Installation

```
npm install --save styled-components
```

Check In `package.json`:

```
{
 "resolutions": {
   "styled-components": "^5"
 }
```

```
}
```

```
import styled from 'styled-components';
```

```
<Title>
      <h1>Hello Raj</h1>
</Title>
```

```
<MyButton></MyButton>
```

```
<Button4 primary>Button4</Button4>
 <Button4>Button4</Button4>
```

```
const Title = styled.h1`
  font-size: 1.5em;
  text-align: center;
  color: palevioletred;
`;

const MyButton = styled.button`
    border:2px solid palevioletred;
    font-weight:bolder;
    border-radius:10px;
    padding:10px 35px;
    background:none;
```

```
    color:skyblue;
    display:block;
    margin:0 auto;
    &:hover{
        background : palevioletred;
        color:white;
    }
`
const Button1 = styled.button`
    font-weight:bolder;
    border-radius:10px;
    padding:10px 35px;
    background:red;
    color:skyblue;
    display:block;
    margin:0 auto;
`
const Button2 = styled(Button1)`
    background:green;
`
const Button3 = styled(Button1)`
    background:blue;
```

`

```
const Button4 = styled.button`
  /* Adapt the colors based on primary prop */
  background: ${props => props.primary ? "palevioletred" : "white"};
  color: ${props => props.primary ? "white" : "palevioletred"};
  font-size: 1em;
  margin: 1em;
  padding: 0.25em 1em;
  border: 2px solid palevioletred;
  border-radius: 3px;
`;
```

===
==========

## Props :

Props is special keywords in React like properties/parameter/argument

Props are arguments passed into React components.

Props are passed to components via HTML attributes.

`props` stands for properties.

React Props are like function arguments in JavaScript *and* attributes in HTML.

To send props into a component, use the same syntax as HTML attributes:

## Example

Add a "brand" attribute to the Car element:

```
const myElement = <Car brand="Ford" />;
```

**Note :always pass parent to child because its one way so we create dynamic component and also all props in form of object**

App.js

```
import Title from "./Component/Title"

function App(){
        return(
        <Title name="Raj" age="20"/>
        <Title name="Akash" age="30"/>
)}
```

Title.jsx

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

class Title extends Component
{
  constructor(props){
        super(props);
        this.props=props;
}

   render(){
        return <h1>I am {this.props.name}, {this.props.age} years</h1>;
   }
}
```

export default Title

**Props in Function component**

**App.js**

```
const demo = (props) =>{
//console.log(props)
return(
        <i>{props.number}</i>
)}
```

Or

Title.jsx

```
function Title()
{
return(
        <App number="1"/>
        <App number="2"/>
        <App number="3"/>
)}
```

Second Method declare props

```
function List({title,desc,img}){
  return (
        <div>
        My title is : {title} and {desc}
        </div>
  )}
```

An use

```
<Myapp title=”Puspa” desc=””/>
```

==========

# React Tailwind

==========

**React State**

- React components have a built-in state object.
- The state object is where you store property values that belong to the component so only accessible inside the component.
- State is mutable , it can be change as per the need
- When the state object changes, the component re-renders
- use inside class Comp: "this.state" is used // Class Component
- use in Functional "useState" hook is used  // func comp

.

Creating the state Object
The s**tate object is initialized** in the constructor:

**this.state = {**
**brand: "Ford",**
**Model:”i10”**
**};**

The state object can contain as **many properties** as you like:

```
constructor()
{
  super();
  this.state = {
     name: "Nagar",
     age: "31"
    }
};

render(){
       <h1>
       Hi {this.state.name}, my age is {this.state.age}
       </h1>
}
```

Refer to the state object anywhere in the component
by using the this.state.propertyname syntax:

**<h1>My {this.state.brand}</h1>**

To change a value in the state object, use the **this.setState()** method.

```
constructor()
{
  super();
  this.state = {
```

```
    name: "Nagar",
    age: "31"
   }
};
changeState = () => {
   console.log(this.state);
   this.setState({name: "Raj", age: "32"}, () => {
            console.log(this.state);
      });
  }

render()
{
      <div>
            <h1>
            Hi {this.state.name}, my age is {this.state.age}
            </h1>
            <br/>

            <button onClick="{() => this. changeState ()}"> Click Me
</button>
      </div>
}
```

**State In Func Component**

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

```
 const [name, setName] = useState("Raj");
    const [age, setAge] = useState(31);
    const [number, setNumber] = useState(0);
    const [img, setImg] = useState(true);
```

```
 <h1>My name is {name} and my age is {age}</h1>
            <button onClick={() => {
                setName("Raj Nagar");
                setAge(32);
            }}>onChange</button>
```

==========

## Event Handling

**Event Binding By () => {}  /  () =>  arrow operator**
make function

```
changeState = () => {
    console.log(this.state);
    console.log(this.props);
  }
```

call function in Class Component

```
render()
{
        <div>
        <button onClick="{this. changeState ()}"> Click Me </button>
        </div>
}
```

call function in FuncComponent

```
render()
{
        <div>
        <button onClick="{changeState}"> Click Me </button>
        </div>
}
```

==========

**Function with argument**

if make function with argument then you have to use ()=>

```
function display2(argument){
    alert(argument);
}
```

```
  <button onClick={()=>this.display2('Rajnagar')}>Call Function in class</button>
```

```
  <button onClick={()=>display2('Rajnagar')}>Call Function in Func</button>
```

==========

**if you go direct function then you got error you have bind event => because you can not use this without arrow function => this provide all**

**Event Binding**

```
changeState(){
   console.log(this.state);
   console.log(this.props);
  }
```

call function

```
render()
{
       <div>
       <button onClick="{() => this. changeState ()}"> Click Me </button>
       </div>
}
```

**Inline Binding**

```
<button onClick="{this. changeState.bind(this)}"> call func in Class </button>

<button onClick="{changeState.bind(this)}"> call func in Func </button>
```

```
constructor(){}  // run before component create

construtor()

{

super() // javascript function for auto call constructor

console.warn("constructor called");

}
```

**Component LifeCycle In React in Class Component**

1) **componentDidMount()**{} // birth// run when component ready to use

=>Means Component Call means mount / use

2) **componentDidUpdate(){}** // marriage// run when any update in compo

=>Means Component setState means state update or Update

3) **componentWillUnmount(){**}// end // run when component remove

=>Means Component Remove from screen means end

Example

```
constructor(){
    super();
    this.state={
       isshowing: false
     }
 }

render() {
   return (
     <div className='container'>
          <button onClick={()=>this.setState({isshowing:true})}>Show</button>
          <button onClick={()=>this.setState({isshowing:false})}>Hide</button>
          <button
onClick={()=>this.setState({isshowing:!this.state.isshowing})}>Hide/show</button>

        {
        this.state.isshowing ? <ImgComp/> : null
        }
     </div>
   )
 }
```

**Life Cycle in Func Compo**

new hooks introduced **useEffect() work as three life cycle**

Didmount

```
useEffect(()=>{
    console.log('didmount');
  },[]);
```

**DidUpdate**

```
useEffect(()=>{
    console.log('didmount/Update');
  },[number]);
```

**WillUnmount**

```
useEffect(()=>{
      return()=>{console.log('Component destroy');} // work when component
destroy
  },[]);
```

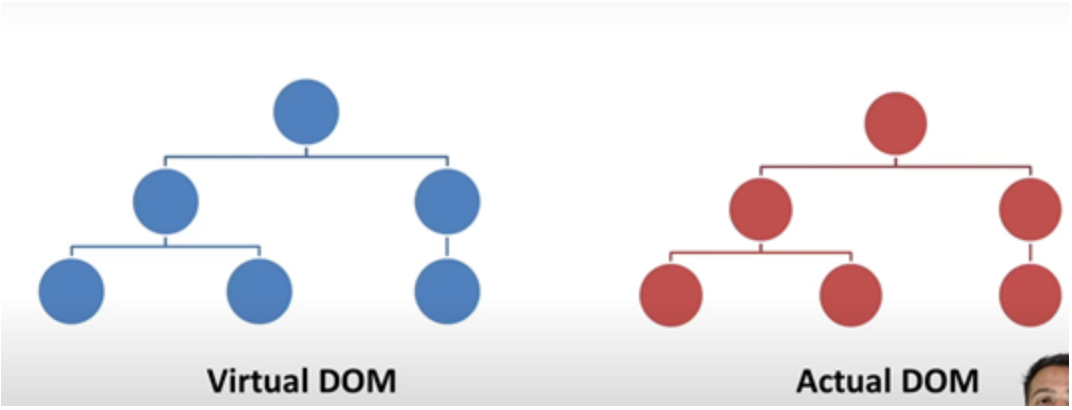
==========

---

Rendering DOM

Virtual Dom

Actual Dom




For only fast render component
Key={item.id}

But id not available in object then :
key={index}

## Rendering the list of Data

### Display Data From object in HTML

OBJEC.JS

```
export const movieData = [
   {
      id: 0,
      title: "pushpa",
      desc: "A labourer named Pushpa makes enemies as he rises in the world of red sandalwood smuggling. However, violence erupts when the police attempt to bring down his illegal business."
   },
   {
      id: 1,
      title: "KG5F",
      desc: "A labourer named Pushpa makes enemies as he rises in the world of red sandalwood smuggling. However, violence erupts when the police attempt to bring down his illegal business."
   }]
```

**Main.jsx**

```
import { movieData } from './Object'

Function Data(){
return(
<div>
        {
            movieData.map((item)=>{
                //return < Object_data title={item.title} desc={item.desc}  />
                    return <Object_data key={item.id}{...item}/>
        /* // if id not available then provide index
        movieData.map((item,index)=>{
                //return < Object_data title={item.title} desc={item.desc}  />
                    return <Object_data key={index}{...item}/>
        */
           })
        }
</div>
)}

Rendering DOM
```

## Display Data from Api in HTML

```
constructor()
  {
      fetch('https://jsonplaceholder.typicode.com/posts')
      .then(response => response.json())
      .then(json => console.log(json))
      .catch(err => console.log(err));
      super();
  }
```

```
const [listdata,setlistdata]=useState([]);
const [isload, SetIsload]=useState(true);
    function list(data)
    {
        setlistdata(data)
```

```
    }
    useEffect(()=>{
        fetch('https://fakestoreapi.com/products')
        .then(res=>res.json())
        .then(data=> {
            SetIsload(false)
            list(data)}
            );
    },[])

  return (
       <>
        {
        isload ?
        <div class="d-flex justify-content-center">
            Loading... <div class="spinner-border"></div>
        </div>
        :
        null}
        {
            listdata.map((item,index)=>{
        return(
        <div className="col-lg-4 col-md-6 col-sm-12 p-0 portfolio-item first
mt-5">
            <div className="position-relative overflow-hidden">
                <div className="portfolio-img">
                <img className="w-100" width="100%" height="250px"
src={item.image} alt />
                </div>
                <div className="portfolio-text bg-primary">
                <h4 className="font-weight-bold mb-4">{item.title}</h4>
                <div className="d-flex align-items-center
justify-content-center">
                    <a className="btn btn-sm btn-secondary m-1" href>
                    <i className="fa fa-link" />
                    </a>
                    <a className="btn btn-sm btn-secondary m-1" href={item.image}
data-lightbox="portfolio">
                    <i className="fa fa-eye" />
```

```
                    </a>
                </div>
                </div>
            </div>
            </div> )
            })
        }

        </>
)
```

==========

**React Hooks /**

Hooks are the new feature introduced in the React 16.8 version. It allows you to use state and other React features without writing a class. Hooks are the functions which "hook into" React state and lifecycle features from function components. It does not work inside classes.
Hooks are backward-compatible, which means it does not contain any breaking changes. Also, it does not replace your knowledge of React concepts.

**When to use a Hooks**

If you write a function component, and then you want to add some state to it, previously you do this by converting it to a class. But, now you can do it by using a Hook inside the existing function component.

**Rules of Hooks Exa : const [name,setname] = useState("Rajesh")**

Hooks are similar to JavaScript functions, but you need to follow these two rules when using them. Hooks rule ensures that all the stateful logic in a component is visible in its source code. These rules are:

1. Only call Hooks at the top level

Do not call Hooks inside loops, conditions, or nested functions. Hooks should always be used at the top level of the React functions. This rule ensures that Hooks are called in the same order each time a components renders.

2. Only call Hooks from React functions

You cannot call Hooks from regular JavaScript functions. Instead, you can call Hooks from React function components. Hooks can also be called from custom Hooks.

```
1. import React, { useState } from 'react';
    2.  // Declare a new state variable, which we'll call "count"
    const [name, setName] = useState("Rajesh")
    const [number, setNumber] = useState(0)
    const [data, setData] = useState([
        "Vinny", "Aditi", "Rajesh", "Parth"
        ])
    const [userProfile, setUserProfile] = useState({
        name: "parth",
        city: "surat",
        lastName: "singh",
        })
return (
  <div>
   <p>You clicked {count} times</p>
   <button onClick={() => setCount(count + 1)}> Click me </button>
   <hr/>

   <h3>{name}</h3>
   <button onClick={() => setName("parth")}>Change the user</button>
   <hr />
   <h3>Total Data : {data.length} </h3>

   {
      data.length <= 0 ? <h1>0 Data Found</h1> :
      data.map((item, index) => {
         return <h2 key={index}>{item}</h2>
       })
   }
  <button onClick={() => setData([])}>Clear All Data</button>
  <hr />

  <h1>{number}</h1>
  <button onClick={increment}>+</button>
  <button onClick={() => setNumber(number - 1)}>-</button>

   <h1>{userobj.name}<h1/>
```

```
    </p><button onClick={()=>
Setuser({...userobj,name:"nagar",mobile:"98548877",age:"25"})}>Change Object
All</button>
```

```
  </div>
 );
}
```

==========

# Hooks Effect / after rander Effect

The Effect Hook allows us to perform side effects (an action) in the function components. It does not use components lifecycle methods which are available in class components. In other words, Effects Hooks are equivalent to componentDidMount(), componentDidUpdate(), and componentWillUnmount() lifecycle methods.
Side effects have common features which the most web applications need to perform, such as:

- Updating the DOM,
- Fetching and consuming data from a server API,
- Setting up a subscription, etc.

```
o  /*
o    useEffect(()=> {
o      console.log('Component mounted');
o    },[]);
o
o  /*
o    useEffect(()=> {
o      alert('Hello');
o      console.log('Component updated);
o    },[number1,number2]);
o  */
o
o
o
o  /*
o    useEffect(()=> {
o      console.log('Component mounted');
o      return ()=> { console.log('Component destroy')}
o    },[]);
o  */
o
o  /*
o    useEffect(()=> {
o      console.log('Component mounted');
```

```
o       Alert('update')
o     });
o   */
o
o
```

Stateful & Stateless Component

In React, a *stateful* component is a component that holds some state. *Stateless* components, by contrast, have no state. Note that both types of components can use props.

# Built-in Hooks

Here, we describe the APIs for the built-in Hooks in React. The built-in Hooks can be divided into two parts, which are given below.

**Basic Hooks**

- useState
- useEffect
- useContext

**Additional Hooks**

- useReducer
- useCallback
- useMemo
- useRef
- useImperativeHandle
- useLayoutEffect
- useDebugValue

HOOKS

```
import { useState,useEffect,useNavigate,useParams,useLocation } from 'react-router-dom';
```

# React useMemo Hook

The React useMemo Hook returns a memoized value.

The useMemo Hook only runs when one of its dependencies update.

This can improve performance.

```
//const calculation= expensiveCalculation(count); withaout useMEMO ALL TIME REDURING

    const calculation= useMemo(()=>expensiveCalculation(count),[count])  // WITH useMemo only function call when depedanct count change
```

==========

# Theme Integration

src : asset => admin  and frontends  all copy assets folder in as per project

src : component

src : layouts => admin and frontend

admin : navbar / sidebar / footer /marsterlayout

master.jsx  :

```
<div class="body">

        <Navabar/>
        <div sidebar>
                <sidebar/>
        </div>
        <div mainclass>
        </div>
        <footer/>
</div>
```

# React Routing  / Template

Step 1: Copy and paste all assets folder in React Public Folder

Step 2: Copy all head path files like .js / .css / script in head or body of index.html

Step 3: Create Component folder in React src & create diff comp

Exa: Nav / header / sidebar / mainbar / footer

Capy all small parts of pages and past in all component

Step 4: Also convert in to jsx

Step 5: call into App.jsx

Step 6: Now For Routing add packages

npm install react-router-dom@6

check in package.json file added or not

Step 7:

https://stackblitz.com/github/remix-run/react-router/tree/main/examples/basic?file=src%2FApp.tsx

Important

```
import { Routes, Route, Outlet, Link , NavLink } from "react-router-dom";
```

App.js

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

<BrowserRouter>
      <MainComp/>
</BrowserRouter>
```

Main.jsx

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


<Routes>
       <Route path="/" element={<Layout />}>
```

```
        <Route index element={<Home />} />
        <Route path="about" element={<About />} />
        <Route path="dashboard" element={<Dashboard />} />

            {/* Using path="*"" means "match anything", so this route
              acts like a catch-all for URLs that we don't have explicit
              routes for. */}

        <Route path="*" element={<NoMatch />} />
      </Route>
    </Routes>
```

Step 8: Define all links in header or navbar by Link/NavLink

```
    import { Link, NavLink} from 'react-router-dom';


      <li>
        <Link to="/">Home</Link>
      </li>
      <li>
        <Link to="/about">About</Link>
      </li>
      <li>
        <Link to="/dashboard">Dashboard</Link>
      </li>
      <li>
        <Link to="/nothing-here">Nothing Here</Link>
      </li>
```

React Router

```
import { BrowserRouter,HashRouter  } from "react-router-dom";
```

COMPONENTS

```
import {Routes,Route, Link,NavLink ,Navigate,Outlet } from "react-router-dom";
```

==========

# React Helmet

React Helmet can be termed as the document **head manager** for React-based applications. Using it, it becomes very easy for developers to update meta tags present on the server-side and the client-side. This library can be termed perfect for applications where SEO plays a crucial role. It is also a media-friendly library with different dependencies and features that make its usage easy in any React app. We will study all the aspects in this tutorial and see how you can add React Helmet into your project.

1. npm install react-helmet

   import { Helmet } from "react-helmet";  in app.js

1. <Helmet>
2. <html lang="en" />
3. <title>React Helmet Tutorial</title>
4. <meta name="description" content="Tutorial for React Helmet" />
5. <meta name="theme-color" content="#E6E6FA" />
6. </Helmet>

==========

API Using by FETCH / AXIOS

<!--

axios

// get all data

```
const fetchall=async()=>{

    const res=await
axios.get(`https://jsonplaceholder.typicode.com/photos`)

}



// get single data by id



const fetchsingle=async()=>{

    const res=await
axios.get(`https://jsonplaceholder.typicode.com/photos/1`)

}




// get particular column data by column



const fetchcolumn=async()=>{

    const res=await
axios.get(`https://jsonplaceholder.typicode.com/photos?category=men`)

}




// delete data by id
```

```
        const delete=async()=>{

                const res=await
axios.delete(`https://jsonplaceholder.typicode.com/photos/2`)

         }




       // Insert data


        const object={

                        albumId: 5001,

                        id: 5001,

                        title: "reprehenderit est deserunt velit ipsam",

                        url: "https://via.placeholder.com/600/771796",

                        thumbnailUrl: "https://via.placeholder.com/150/771796"

                      }


        const insert=async()=>{

                const res=await
axios.post(`https://jsonplaceholder.typicode.com/photos`,object)

         }
```

```
        // update data


         const object={
                        albumId: 5001,
                        id: 5001,
                        title: "reprehenderit est deserunt velit ipsam",
                        url: "https://via.placeholder.com/600/771796",
                        thumbnailUrl: "https://via.placeholder.com/150/771796"
                       }


         const update=async()=>{
                const res=await
axios.patch(`https://jsonplaceholder.typicode.com/photos/2`,object)
          }


//=============================================
=========================================


// fetch
// get all data


         const fetchdata = () => {
```

```
    fetch('https://jainam-3b289-default-rtdb.firebaseio.com/users')
        .then((response) => response.json())
        .then((data) => { console.log(data)});
}


    // get single data by id


    const fetchsingle = () => {
    fetch('https://jainam-3b289-default-rtdb.firebaseio.com/users/2')
        .then((response) => response.json())
        .then((data) => { console.log(data)});
}


    // get particular column data by column


    const fetchcolumn = () => {
    fetch('https://jainam-3b289-default-rtdb.firebaseio.com/users?name=raj')
        .then((response) => response.json())
        .then((data) => { console.log(data)});
}
```

```
    // delete


    const ondelete =()=> {
   fetch(`https://jainam-3b289-default-rtdb.firebaseio.com/users/2`, {
       method: 'DELETE',
   })
            .then((response) => {


            })
}



    // inset data


    const insert = () => {
   fetch(`https://jainam-3b289-default-rtdb.firebaseio.com/users`, {
       method: 'POST',
       body: JSON.stringify(object),
       headers: {
           'Content-type': 'application/json; charset=UTF-8',
       },
   })
```

```
    .then((response) => response.json())
    .then((object) =>{ console.log(object) });
 }




 // update data


 const onupdate = () => {
fetch(`https://jainam-3b289-default-rtdb.firebaseio.com/users/1`, {
    method: 'PATCH',
    body: JSON.stringify(object),
    headers: {
        'Content-type': 'application/json; charset=UTF-8',
    },
})
    .then((response) => response.json())
    .then((object) =>{ console.log(object) });
 }
 -->
```

==========

Form Handling // Get Data From

Handling in React

● Value ● Onchange

To get the value from the user we use useState and we have to add to value attribute to a input tag onChange is a event listener will fire every time when we change the input value

look at the starter code below ● This Form component has 5 input fields in total; 5 different states and 5 different onChange inline functions ● This is not exactly DRY code

Method:1

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


function Form_handeling() {
  const [name, setname] = useState("");
  const [email, setemail] = useState("");
  const [subject, setsubject] = useState("");
  const [message, setmessage] = useState("");

  const [alldata, setalldata] = useState([])


  function buttonhandel(e) {
    e.preventDefault()
```

```
    const formvalue = { name: name, email: email, subject: subject, message:
message }

    setalldata([...alldata, formvalue]);

    //console.log(alldata);

    setname("");

    setemail("");

    setsubject("");

    setmessage("");

  }




  return (

    <div>

      <div className="container-fluid py-5">

        <div className="container">

          <div className="row align-items-end mb-4">

            <div className="col-lg-6">

              <h6 className="text-secondary font-weight-semi-bold text-uppercase
mb-3">Contact Form</h6>

            </div>

            <div className="col-lg-6">

              <h6 className="text-secondary font-weight-semi-bold text-uppercase
mb-3">Contact Data</h6>

            </div>

          </div>

          <div className="row">
```

```
            <div className="col-lg-6 mb-5 mb-lg-0">

              <div className="contact-form">

                <div id="success" />

                <form name="sentMessage" id="contactForm" >

                  <div className="form-row">

                    <div className="col-sm-6 control-group">

                      <input type="text" name="name" value={name} onChange={(e)
=> setname(e.target.value)} className="form-control p-4" id="name"
placeholder="Your Name" required="required"
data-validation-required-message="Please enter your name" />

                      <p className="help-block text-danger" />

                    </div>

                    <div className="col-sm-6 control-group">

                      <input type="email" name="email" value={email}
onChange={(e) => setemail(e.target.value)} className="form-control p-4"
id="email" placeholder="Your Email" required="required"
data-validation-required-message="Please enter your email" />

                      <p className="help-block text-danger" />

                    </div>

                  </div>

                  <div className="control-group">

                    <input type="text" name="subject" value={subject}
onChange={(e) => setsubject(e.target.value)} className="form-control p-4"
id="subject" placeholder="Subject" required="required"
data-validation-required-message="Please enter a subject" />

                    <p className="help-block text-danger" />

                  </div>

                  <div className="control-group">

                    <textarea name="message" value={message} onChange={(e) =>
setmessage(e.target.value)} className="form-control p-4" id="message"
```

```
placeholder="Message" required="required"
data-validation-required-message="Please enter your message" defaultValue={""} />

                    <p className="help-block text-danger" />

                  </div>

                  <div>

                    <button onClick={buttonhandel} className="btn btn-primary
btn-block py-3 px-5" type="submit" id="sendMessageButton">Send Message</button>

                  </div>

                </form>

              </div>

            </div>

            <div className='col-md-6'>

              <table className='table'>

                <tr>

                  <th>Id</th>

                  <th>Name</th>

                  <th>Email</th>

                  <th>Subject</th>

                  <th>Message</th>

                </tr>

                {

                  alldata.map((item, index) => {

                    //const {id,fname,email} =item

                    return (

                      <tr key={index}>

                        <td>{index + 1}</td>
```

```
                        <td>{item.name}</td>
                        <td>{item.email}</td>
                        <td>{item.subject}</td>
                        <td>{item.message}</td>
                      </tr>
                    )
                  })

                }

              </table>

            </div>
          </div>
        </div>
      </div>
    </div>
  )
}
export default Form_handeling
```

Method: 2

```
import React,{useState} from 'react'
function Form_main() {
  const[formvalue,setformvalue]=useState({
      name:"",
```

```
    email:"",
    subject:"",
    message:""
  })
  const[alldata,setalldata]=useState([]);

  function changehandel(e){
    setformvalue({...formvalue,[e.target.name]:[e.target.value]})
  }
  function submithandel(e){
    e.preventDefault();
    setalldata([...alldata,formvalue]);
    setformvalue({name:"",email:"",subject:"",message:""})
  }

  return (
    <div>
      <div className="container-fluid py-5">
        <div className="container">
          <div className="row align-items-end mb-4">
            <div className="col-lg-6">
              <h6 className="text-secondary font-weight-semi-bold text-uppercase
mb-3">Contact Form</h6>
            </div>
            <div className="col-lg-6">
```

```
              <h6 className="text-secondary font-weight-semi-bold text-uppercase
mb-3">Contact Data</h6>

            </div>

          </div>

          <div className="row">

            <div className="col-lg-6 mb-5 mb-lg-0">

              <div className="contact-form">

                <div id="success" />

                <form name="sentMessage" id="contactForm" >

                  <div className="form-row">

                    <div className="col-sm-6 control-group">

                      <input type="text" name="name" value={formvalue.name}
onChange={changehandel} className="form-control p-4" id="name" placeholder="Your
Name" required="required" data-validation-required-message="Please enter your
name" />

                      <p className="help-block text-danger" />

                    </div>

                    <div className="col-sm-6 control-group">

                      <input type="email" name="email" value={formvalue.email}
onChange={changehandel} className="form-control p-4" id="email" placeholder="Your
Email" required="required" data-validation-required-message="Please enter your
email" />

                      <p className="help-block text-danger" />

                    </div>

                  </div>

                  <div className="control-group">

                    <input type="text" name="subject" value={formvalue.subject}
onChange={changehandel} className="form-control p-4" id="subject"
placeholder="Subject" required="required"
data-validation-required-message="Please enter a subject" />
```

```
                    <p className="help-block text-danger" />

                  </div>

                  <div className="control-group">

                    <textarea name="message" value={formvalue.message}
onChange={changehandel} className="form-control p-4" id="message"
placeholder="Message" required="required"
data-validation-required-message="Please enter your message" defaultValue={""} />

                    <p className="help-block text-danger" />

                  </div>

                  <div>

                    <button onClick={submithandel} className="btn btn-primary
btn-block py-3 px-5" type="submit" id="sendMessageButton">Send Message</button>

                  </div>

                </form>

              </div>

            </div>

            <div className='col-md-6'>

              <table className='table'>

                <tr>

                  <th>Id</th>

                  <th>Name</th>

                  <th>Email</th>

                  <th>Subject</th>

                  <th>Message</th>

                </tr>

                {

                  alldata.map((item, index) => {
```

```
                    //const {id,fname,email} =item
                    return (
                      <tr key={index}>
                        <td>{index + 1}</td>
                        <td>{item.name}</td>
                        <td>{item.email}</td>
                        <td>{item.subject}</td>
                        <td>{item.message}</td>
                      </tr>
                    )
                  })

                }

              </table>

            </div>
          </div>
        </div>
      </div>
    )
}

export default Form_main
```

==========

==========

Api Crud

Firebase

https://firebase.google.com/

https://firebase.google.com/docs/reference/rest/database Doc Guide

Insert

post : https://reactcrud-576db-default-rtdb.firebaseio.com/users.json

peste in postmen

go in row menu and add column & value & select json format

```
{
    "name":"rajesh",
    "email":"rajesh@gmail.com",
    "subject":"demo",
    "message":"hello how are you"
}
```

Get Data

Get: https://reactcrud-576db-default-rtdb.firebaseio.com/users.json

Update Data

patch: https://reactcrud-576db-default-rtdb.firebaseio.com/users/id.json

select one id & past in ural & then also want to update data then again go in body /row & select json and

{

‘name’ :’updated name’

}

Delete

delete: https://reactcrud-576db-default-rtdb.firebaseio.com/users/id.json

https://firebase.google.com/docs/reference/rest/database/

Postmen

==========

Crud in API from api placeholder guide

https://jsonplaceholder.typicode.com/guide/

Insert Data:

```
function Api_crud() {
  const InitialValue={
    fname:"",
    email:""
}
const [formValue,SetFormvalue]=useState(InitialValue);
function handelchange(e){
  //SetFormvalue({...formValue,[e.target.name]:[e.target.value]})
  const { name, value}=e.target
  SetFormvalue({...formValue,[name]:value});
}
function handelSubmit(e)
{
     e.preventDefault();
     fetch(`https://reactcrud-576db-default-rtdb.firebaseio.com/contacts.json`, {
        method: 'POST',
        body: JSON.stringify(formValue),
        headers: {
            'Content-type': 'application/json; charset=UTF-8',
        },
})
.then((response)=>response.json())
.then((response) => {
    if(response)
    {
        SetFormvalue({fname:"",email:""});
        getData();
    }
```

```
        <input type="text"
            placeholder='email'
            className='form-control mb-3'
            name='email'
           value={formValue.email}
           onChange={handelchange}
        />
        <button className='btn btn-primary' onClick={handelSubmit}>Submit</button>
    </form>
    <br></br>
    <table className='table'>
        <tr>
            <th>Id</th>
            <th>Name</th>
            <th>Email</th>
        </tr>
        {
            Object.keys(alldata).map((item,index) => {
                //const {fname,email} =item
                return(
                    <tr key={index}>
                        <td>{index +1}</td>
                        <td>{alldata[item].fname}</td>
                        <td>{alldata[item].email}</td>
                    </tr>
                )
            })
```

Get Data:

```
const [alldata,setalldata]=useState("");

function getData(){
fetch("https://reactcrud-576db-default-rtdb.firebaseio.com/contacts.json")
.then((response) => response.json())
.then((data) => setalldata(data));
}

useEffect(()=>{
    getData()
},[])

/*
let data={id:1,name:"Raj"}
console.log(data);
console.log(data.id)// (data["id"])
console.log(Object.keys(data));
Object.keys.map((item)=>{console.log(item)})// console.log(data[item])
*/
```

```
{
    Object.keys(alldata).map((item,index) => {
        //const {fname,email} =item
        return(
            <tr key={index}>
                <td>{index +1}</td>
                <td>{alldata[item].fname}</td>
                <td>{alldata[item].email}</td>
                <td><button class="btn btn-danger" onClick={()=>handelDelete(item)}>Delete<
            </tr>
        )
    })

}
```

Delete Data :

```
function handelDelete(deleteid)
{
    let ans=window.confirm("Are you sure want to delete !");
    if(ans)
    {
        //alert(deleteid);
    fetch(`https://reactcrud-576db-default-rtdb.firebaseio.com/contacts/${deleteid}.json`, {
        method: 'DELETE',
        })
        .then((response) => response.json())
        .then((data) => {
            if(data === null)
            {
                getData();
            }
        });
    }

}
```

# UseContext :

React Context is a way to manage state globally.

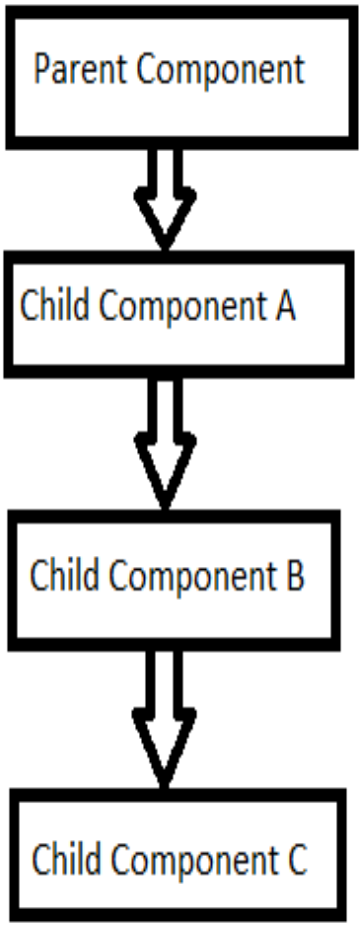
It can be used together with the useState Hook to share state between deeply nested components more easily than with useState alone.

# The Problem / Props Drilling

State should be held by the highest parent component in the stack that requires access to the state.

To illustrate, we have many nested components. The components at the top and bottom of the stack need access to the state.

To do this without Context, we will need to pass the state as "props" through each nested component. This is called "**prop  drilling**" .



https://www.w3schools.com/react/react_usecontext.asp

Even though components 2-3-4 did not need the state, they had to pass the state along so that it could reach component 5.

# UseContex / CreateContext

The solution is to create context.  To create context and use Context, you must Import `createContext / useContext` and initialize it:

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

const UserContext = createContext()

Next we'll use the Context Provider to wrap the tree of components that need the state Context.

Context Provider : Wrap child components in the Context Provider and supply the state value.

```
function Component1() {
  const [user, setUser] = useState("Jesse Hall");

 const UserContext = createContext()

  return (
    <UserContext.Provider value={user}>

      <h1>{`Hello ${user}!`}</h1>
      <Component2 user={user} />

    </UserContext.Provider>
  );
}
```

Now, all components in this tree will have access to the user Context.

# Use the `useContext` Hook

In order to use the Context in a child component, we need to access it using the `useContext` Hook.First, include the `useContext` in the import statement:

```
import { useState, createContext, useContext } from "react";
import ReactDOM from "react-dom/client";

const UserContext = createContext();

function Component1() {
const [user, setUser] = useState("Jesse Hall");

  return (
    <UserContext.Provider value={user}>
      <h1>{`Hello ${user}!`}</h1>
      <Component4 user={user} />
    </UserContext.Provider>
  );
}
function Component4() {
  return (
    <>
      <h1>Component 4</h1>
      <Component5 />
    </>
  );
}

function Component5() {
  const user = useContext(UserContext);

  return (
    <>
      <h1>Component 5</h1>
      <h2>{`Hello ${user} again!`}</h2>
    </>
  );
}
```

==========

# Json-Server

JSON Server is a lightweight and easy-to-use Node. js tool that simulates a RESTful API using a JSON file as the data source. With JSON Server, front-end developers can create mock APIs without the need to write complex server-side code, or when a backend API isn't ready yet.

step : 1 Json Server install package / For api

**npm install -g json-server**

step : 2

=> create **db.json** in root for json server api

add data in json type

```
{
    "user": [
      {
        "id": 1,
        "name": "demo",
        "email": "demo@gmail.com",
        "password": "123456",
        "mobile": 9722041171,
        "status": "Unblock"
      },{
        "id": 2,
        "name": "demo",
```

```
      "email": "demo@gmail.com",
      "password": "123456",
      "mobile": 9722041171,
      "status": "Unblock"
    }

  ],
  "contact": [
    {
      "name": "jay",
      "email": "jay@gmail.com",
      "sub": "demo",
      "msg": "hi",
      "contactid": "1679640569760",
      "id": 1
    }
  ],
  "admin": [
    {
      "id": 1,
      "name": "admin",
      "email": "admin@gmail.com",
      "password": "123456"
    }
  ]
}
```

step:3
=> add in package.json  script / run db.json file in server

```
    "server": "json-server --watch db.json --port 3000"
```

or

run locally then use

Step : 4

npx json-server --watch db.json

http://localhost:3000/user  all data

http://localhost:3000/user/1  get id=1 data

localhost:3000/user?name=demo  get particular column data

==========

## Postman

**=> download & Install**

**=> Signup with gmail**

**=> create workplace**

**=> In workplace create Collection**

**=> In Collection create Re+quest for testing API**

### GET - Reading Data

**http://localhost:3000/user  all get**

**http://localhost:3000/user/2  single get**

### POST - Pushing Data

**Post :http://localhost:3000/user**

**changes :**

**header: accept : application/json**

**Body : => raw  => set json**

```
{
   "id": 3,
   "name": "demo 3",
   "email": "demo@gmail.com",
   "password": "123456",
```

```
  "mobile": 9722041171,
  "status": "Unblock"
}
```


## PATCH - Updating Data

http://localhost:3000/user/1 **update**


```
{
  "name":"Raj Nagar"
}
```


## DELETE - Removing Data

http://localhost:3000/user/3 **Delete**

==========

Sweet alert react / https://sweetalert.js.org/guides/

```
npm install sweetalert --save
```

```
import swal from 'sweetalert';

swal({
  title: "Good job!",
  text: "You clicked the button!",
  icon: "success",
  button: "Aww yiss!",
});
```

Axios :

Axios is **a lightweight HTTP client based on the $http service within Angular. js v1. x** and is similar to the native JavaScript Fetch API. Axios is promise-based, which gives you the ability to take advantage of JavaScript's async and await for more readable asynchronous code.

Install : npm install axios

Then you can use directly in component

Import axios from ‘axios’

Now we can work with api exa: jsoneplaceholder : fake api
https://jsonplaceholder.typicode.com/

Insert

```
const initialiation = {
        name: "",
        email: "",
        password: "",
    }
    const [formvalue, setformvalue] = useState(initialiation);

    function onchangevalue(e) {
        const { name, value } = e.target
        setformvalue({ ...formvalue, [name]: value });

    }
    const [err_msg,setErr_msg]=useState([]);
    const submitHandel = async (e) => {
        e.preventDefault();
        const res=await
axios.post(`http://localhost:8000/api/insertstudent`,formvalue);
        if(res.data.status === 200)
        {
            //alert(res.data.msg);
            swal({
                title: "Success",
                text: res.data.message,
                icon: "success",
                button: "Ok!",
              });
```

```
            setformvalue({...formvalue,name:"",email:"",password:""})
            setErr_msg([]);
            //console.log(res.data.msg);
        }
        else
        {
            setErr_msg(res.data.message)
            //console.log([err_msg]);
        }
    }
```

view Form

```
<label htmlFor="name">Name:</label>

<input type="text" className="form-control" value={formvalue.name}
onChange={onchangevalue} id="email" />

<span className='text-danger'>{err_msg.name}</span>
```

==========

show all

```
import React, { useState, useEffect} from 'react'
import {useNavigate,Link,NavLink} from 'react-router-dom'
import axios from 'axios';
import swal from 'sweetalert';

function Index() {

    useEffect(() => {
        fetchall();
    }, [])

    const [alldata, setAlldata] = useState([]);
    const [isload,setisload]=useState(true);

async function fetchall() {
        const res=await axios.get(`http://localhost:8000/api/student`);
```

```
        //console.log(res);
        if(res.data.status === 200)
        {
            setAlldata(res.data.students);
            setisload(false);
        }
    }
```

View

```
 var tabledata_HTML="";
    if(isload)
    {
        tabledata_HTML= <tr><td><h1>Data Loading...</h1></td></tr>
    }
    else
    {
        tabledata_HTML = alldata.map((item, index) => {

            return (
                <tr key={item.id}>
                    <td>{item.id}</td>
                    <td>{item.name}</td>
                    <td>{item.email}</td>
                    <td><button className='btn btn-primary text-primary'
data-bs-toggle="modal" data-bs-target="#myModal" onClick={() =>
handeledit(item.id)}>Edit</button></td>
                    <td><button className='btn btn-danger text-danger'
onClick={() => handelDelete(item.id)}>Delete</button></td>
                    <td><button className='btn btn-primary text-primary'
onClick={() => handelstatus(item.id)}>{item.status}</button></td>
                </tr>
            )
        })
    }
```

View Table

```
{tabledata_HTML}
```

==========

edit / single show

```
const initialiation = {
        name: "",
        email: "",
        password: ""
    }
    const [editformvalue, seteditformvalue] = useState(initialiation);
    const [id,setId]=useState({});
   async function handeledit(EditID)
    {
        const res= await
axios.get(`http://localhost:8000/api/student/${EditID}`);
        if(res.data.status === 200)
        {
            seteditformvalue(res.data.students);
            setId(EditID);
        }
    }
```

==========

```
// update

    function onchangeeditvalue(e) {
        const { name, value } = e.target
        seteditformvalue({ ...editformvalue, [name]: value })
        //console.log(editformvalue);

    }

 const [err_msg,setErr_msg]=useState([]);

async function handelupdate(e) {

e.preventDefault()
const res= await
axios.put(`http://localhost:8000/api/updatestudent/${id}`,editformvalue);
        if(res.data.status === 200)
        {
            //alert(res.data.msg);
```

```
            swal({
                title: "Success",
                text: res.data.msg,
                icon: "success",
                button: "Ok!",
              });
            fetchall();
            setErr_msg([]);
        }
        else
        {
            setErr_msg(res.data.message)
        }
    }
```

==========

delete

```
async function handelDelete(deleteid) {
const  res= await
axios.delete(`http://localhost:8000/api/student/${deleteid}`);
        if(res.data.status === 200)
        {
            //alert(res.data.msg);
            swal({
                title: "Success",
                text: res.data.msg,
                icon: "success",
                button: "Ok!",
              });
            fetchall();
        }
    }
```

==========

Update Status

```
async function handelstatus(statusid) {
```

```
 const  res= await
axios.put(`http://localhost:8000/api/updatestatus/${statusid}`);
        if(res.data.status === 200)
        {
           //alert(res.data.msg);
           swal({
              title: "Success",
              text: res.data.msg,
              icon: "success",
              button: "Ok!",
            });
           fetchall();
        }
    }
```

Login with firebase

https://firebase.google.com/docs/reference/rest/auth

```
//Login

const initialiation = {
        email: "",
        password: "",
    }
    const [formvalue, setformvalue] = useState(initialiation);

    function onchangevalue(e) {
        const { name, value } = e.target
        setformvalue({ ...formvalue, [name]: value });

    }

    const navigat=useNavigate();
    const [err_msg,setErr_msg]=useState([]);
    const submitHandel = async (e) => {
        e.preventDefault();
```

```
const res=await axios.post(`http://localhost:8000/api/login`,formvalue);
if(res.data.status === 200)
{
   // store data in local store
   localStorage.setItem('auth_token',res.data.token)
   localStorage.setItem('auth_token',res.data.token)
   //alert(res.data.msg);
   swal({
      title: "Success",
      text: res.data.msg,
      icon: "success",
      button: "Ok!",
    });
   setformvalue({...formvalue,email:"",password:""})
   return navigat('/all_data');
   //console.log(res.data.msg);
}
else if(res.data.status === 201)
{
   swal({
      title: "error",
      text: res.data.msg,
      icon: "error",
      button: "error!",
    });
   setformvalue({...formvalue,email:"",password:""})
   setErr_msg([]);
}
else
{
   setErr_msg(res.data.message)
   //console.log([err_msg]);
}
```

Local STORAGE LIKE SESSION

```
// store data in local store
      localStorage.setItem('auth_token',res.data.token)
      localStorage.setItem('auth_name',res.data.token)
```

```
// check & print

if(!localStorage.getItem('auth_token'))
{
}
// Delete / Logout

localStorage.removeItem('auth_token')
localStorage.removeItem('auth_name')
```

==========

useReducer

This Hook is an alternative to useState, and works similarly to the Redux library. We can use it as follows:

import { useReducer } from 'react'

const [ state, dispatch ] = useReducer(function, objectstate, init)

The useReducer Hook is used to deal with complex state logic

The useReducer is a hook I use sometimes to manage the state of the application. It is very similar to the useState hook, just more complex. It acts as an alternate hook to the useState hook to manage complex state in your

application. The useReducer hook uses the same concept as the reducers in Redux. It is basically a pure function, with no side-effects

```
function functreducer(state,action)
{
    if(action.type==="setchange")
    {
       return { ...state,name:"Akash" }
    }
    if(action.type==="plus")
```

```
    {
        return { ...state,number:+state.number+1 }
    }
    if(action.type==="minus")
    {
        return { ...state,number:+state.number-1 }
    }
    if(action.type==="hide")
    {
        return { ...state,isImage:false }
    }
    if(action.type==="show")
    {
        return { ...state,isImage:true }
    }
    if(action.type==="hideshow")
    {
        return { ...state,isImage:!state.isImage }
    }
}
const initialState={
    number:"1",
    name:"Raj",
    isImage:true
    }

const [state,dispatch]=useReducer(functreducer,initialState)
```

```
<button onClick={() => dispatch({ type: "CHANGE" })}>Change Name</button>
            <h1>{state.name}</h1>
```

==========

## Redux : state management library

**Redux is an open-source JavaScript library used to manage application state.** React uses Redux for building the user interface. It was first introduced by **Dan Abramov** and **Andrew Clark** in **2015**.

**React Redux is the official React binding for Redux. It allows React components to read data from a Redux Store, and dispatch Actions to the Store to update data.** Redux helps apps to scale by providing a sensible way to manage state through a unidirectional data flow model. React Redux is conceptually simple. It subscribes to the Redux store, checks

to see if the data which your component wants have changed, and re-renders your component.

Redux was inspired by Flux. Redux studied the Flux architecture and omitted unnecessary complexity.

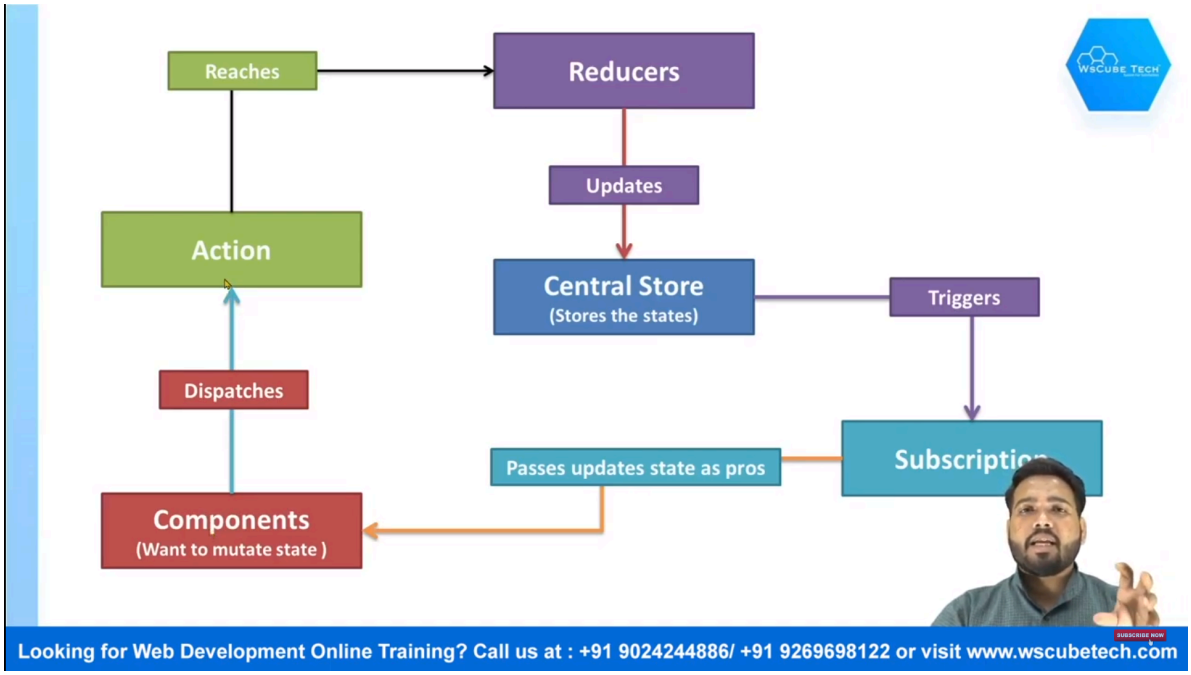
- Redux does not have Dispatcher concept.
- Redux has an only Store whereas Flux has many Stores.
- The Action objects will be received and handled directly by Store.

# Why use React Redux?

The main reason to use React Redux are:

- React Redux is the official **UI bindings** for react Application. It is kept up-to-date with any API changes to ensure that your React components behave as expected.
- It encourages good 'React' architecture.
- It implements many performance optimizations internally, which allows to components re-render only when it actually needs to.

Central store (all state in)



redux work in two ways

1) Redux
2) React-Redux
3) Redux Toolkit

1) Redux

**Redux Core**

The Redux core library is available as a package on NPM for use with a module bundler or in a Node application:

```
npm install redux
```

2) React-redux

# Predictable

Redux helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test.

# Centralized

Centralizing your application's state and logic enables powerful capabilities like undo/redo, state persistence, and much more.

# Debuggable

The Redux DevTools make it easy to trace when, where, why, and how your application's state changed. Redux's architecture lets you log changes, use "time-travel debugging", and even send complete error reports to a server.

# Flexible

Redux works with any UI layer, and has a large ecosystem of addons to fit your needs.

## 3) Redux Toolkit

### Simple

Includes utilities to simplify common use cases like store setup, creating reducers, immutable update logic, and more.

### Opinionated

Provides good defaults for store setup out of the box, and includes the most commonly used Redux addons built-in.

### Powerful

Takes inspiration from libraries like Immer and Autodux to let you write "mutative" immutable update logic, and even create entire "slices" of state automatically.

### Effective

Lets you focus on the core logic your app needs, so you can do more work with less code.

**Install Redux :**

```
npx create-react-app my-app
npm install react-redux
```

```
npx create-react-app my-app --template redux
```

**=> then just check package.json for dependency**

```
"dependencies": {
   "@reduxjs/toolkit": "^1.8.3",
   "@testing-library/jest-dom": "^5.16.4",
   "@testing-library/react": "^13.3.0",
   "@testing-library/user-event": "^14.2.6",
   "react": "^18.2.0",
   "react-dom": "^18.2.0",
   "react-redux": "^8.0.2",
   "react-scripts": "5.0.1",
   "web-vitals": "^2.1.4"
 },
```

**==========**

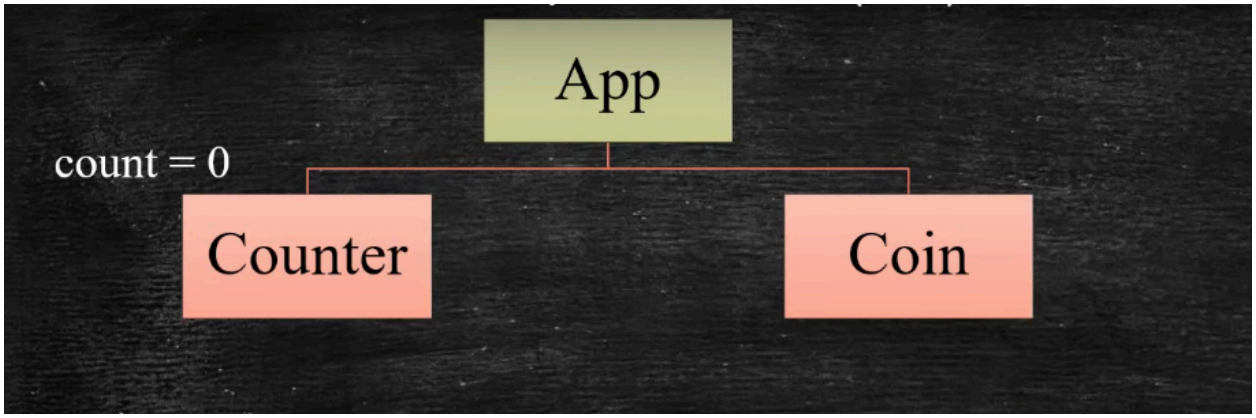
**also some framework folder created in SRC FOLDER**

app : store.js
features :  counter : all counters files

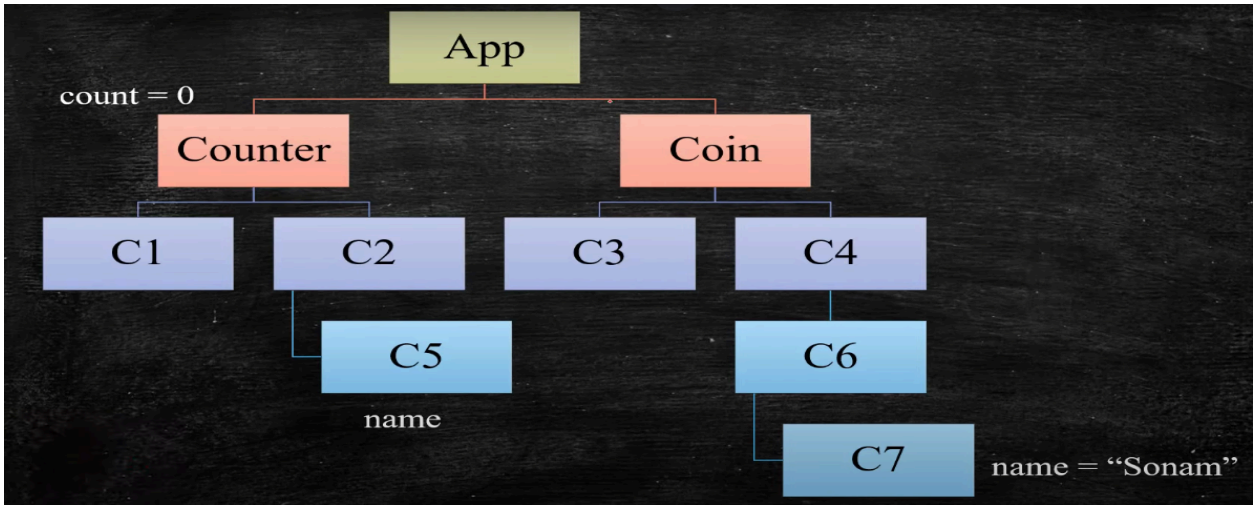
### An Existing React App

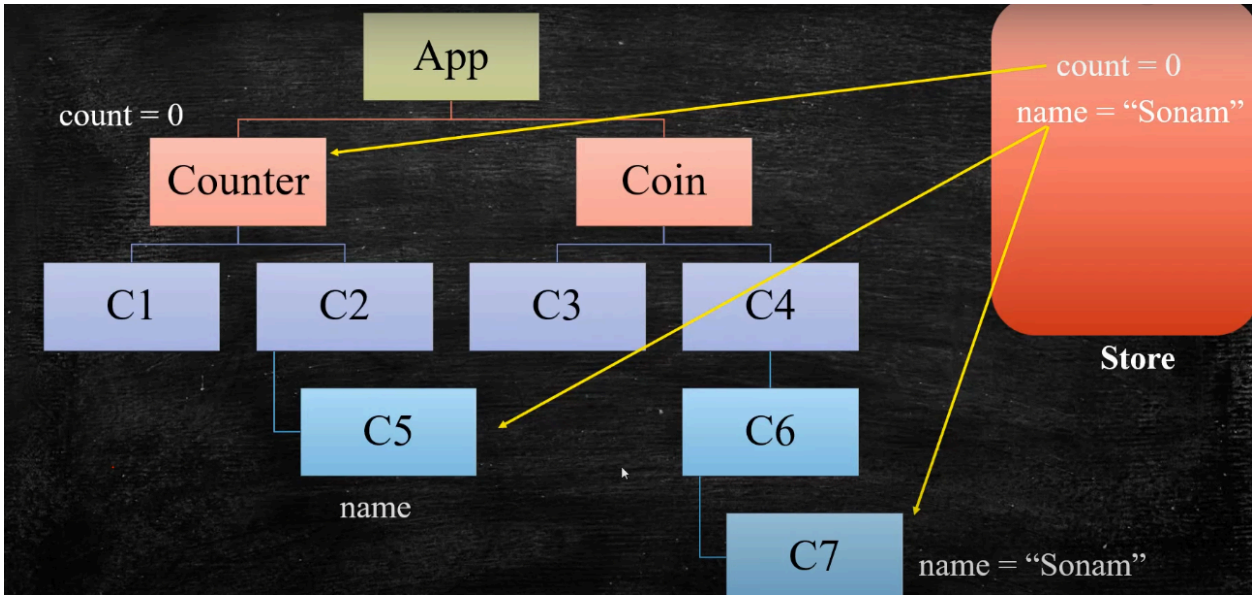
**npx create-react-app my-proj**
**npm install @reduxjs/toolkit**
**npm install react-redux  // install redux for react**

**STATE Management**




**Props Drilling**
**useContext**



https://react-redux.js.org/tutorials/quick-start

Make project in React-redux

step 1: make theme integration as per template & routes

step 2 : create new store in **app/store.js then load in index.js**
step 3 : create new RootReducer  in **app/**rootReducer **.js and load store.js**

**step 4: create actionType then import in all reducer**
**step 5 : features/user/UserReducer  => Create Reducer export into rootReducer**

**step 6: create action for each reducer**

==========

**step 4 : use all state & action in counter.jsx by using**

```
import { useSelector, useDispatch } from 'react-redux'
import { decrement, increment } from './counterSlice'
```

```
onClick={() => dispatch(increment())}
```

==========

# React Redux

step : 1 Theme integration
step: 2 Routing

step : 3 Json Server install package / For api

**npm install -g json-server**

npm start : Run project

=> create **db.json** in root for json server api
add data in json type
=> create **.env** in root for json server api
=> add in package.json script / run db.json file in server

```
"server": "json-server --watch db.json --port 3000"
```

or

run locally then use

npx json-server --watch db.json

http://localhost:3000/users

Step : 4 Redux Start

**npm i redux react-redux @reduxjs/toolkit redux-thunk axios**

**create app folder**

app / store.js
app / rootReducer.js

index.js

```
import store from './app/store';
import { Provider } from 'react-redux';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
```

```
);
```

store.js

```
import { createStore,applyMiddleware } from 'redux'
import reduxThunk from 'redux-thunk'
import rootReducer from './rootReducer';

const middlewares=[reduxThunk];

const store=createStore(rootReducer,applyMiddleware(...middlewares));

export default store;
```

rootReducer.js

```
import { combineReducers } from "redux";
import {userReducer} from '../feature/user/userReducer'
import {} from '..'
const rootReducer=combineReducers({
    user:userReducer,
 })
```

Feature/userReducer

userReducer

```
const statedata={
    alluser:[],
    singleuser:{}
}
const userReducer=(state=statedata,action)=>{
    switch(action){
        default:
        return state;
    }
}
```

```
export default userReducer;
```

actionType.js

```
// create all action what you want on user table

export const GET_USER ="GET_USER";
export const ADD_USER ="ADD_USER";
export const DELETE_USER ="DELETE_USER";
export const UPDATE_USER ="UPDATE_USER";
export const GET_SINGLE_USER ="GET_SINGLE_USER";
```

action.js

**What is a toast notification?**

Toast or Toastify notifications are pop-up messages that display some information to the user. This information could be a success message, warning, error, and so on, as shown in the below:

```
/* NPM */
```

```
$ npm install --save react-toastify
```

step :1 pass in app.js

```
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
```

step :2 pass <ToastContainer/> in app.js

step:3 when you want use add in that component

```
import { toast } from 'react-toastify’
```

```
toast.success('Success Notification !', {
    position: toast.POSITION.TOP_RIGHT
});
toast.error('Error Notification !', {
    position: toast.POSITION.TOP_CENTER
});
```

```
toast.warning('Warning Notification !', {
    position: toast.POSITION.TOP_LEFT
});
toast.info('Information Notification !', {
    position: toast.POSITION.BOTTOM_CENTER
});
toast('Default Notification !', {
    position: toast.POSITION.BOTTOM_LEFT
});
toast('Custom Style Notification with css class!', {
     position: toast.POSITION.BOTTOM_RIGHT,
className: 'foo-bar'
});
```

```
toast.success('Success Notification !', {
    position: toast.POSITION.TOP_RIGHT
});
toast.success('Success Notification !', {
    position: toast.POSITION.TOP_CENTER
});
toast.success('Success Notification !', {
    position: toast.POSITION.TOP_LEFT
});
toast.success('Success Notification !', {
    position: toast.POSITION.BOTTOM_RIGHT
});
toast.success('Success Notification !', {
    position: toast.POSITION.BOTTOM_LEFT
});
toast.success('Success Notification !', {
```

```
    position: toast.POSITION.BOTTOM_CENTER

});
```

# Redux Toolkit

## Create a Redux Store

**npm install @reduxjs/toolkit react-redux**
**npm install redux-devtools-extension –save-dev for extension**

**Create a file named src/app/store.js. Import the configureStore API from Redux Toolkit. We'll start by creating an empty Redux store, and exporting it:**

app/store.js

```
import { configureStore } from '@reduxjs/toolkit'

export default configureStore({
 reducer: {},
})
```

## Provide the Redux Store to React

**Once the store is created, we can make it available to our React components by putting a React Redux <Provider> around our application in src/index.js. Import the Redux store we just created, put a <Provider> around your <App>, and pass the store as a prop:**

index.js

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import './index.css'
import App from './App'
import store from './app/store'
import { Provider } from 'react-redux'
```

```
// As of React 18
const root = ReactDOM.createRoot(document.getElementById('root'))

root.render(
 <Provider store={store}>
   <App />
 </Provider>
)
```

### Create a Redux State Slice

Add a new file named src/features/counter/counterSlice.js. In that file, import the createSlice API from Redux Toolkit.

features/counter/counterSlice.js

```
import { createSlice } from '@reduxjs/toolkit'

export const counterSlice = createSlice({
  name: 'counter',
  initialState: {
    value: 0,
  },  reducers: {

    increment: (state) => {
      // Redux Toolkit allows us to write "mutating" logic in reducers. It
      // doesn't actually mutate the state because it uses the Immer
library,
      // which detects changes to a "draft state" and produces a brand
new
      // immutable state based off those changes
      state.value += 1
    },
    decrement: (state) => {
      state.value -= 1
    },
    incrementByAmount: (state, action) => {
```

```
    state.value += action.payload
   },
  },
})

// Action creators are generated for each case reducer function
export const { increment, decrement, incrementByAmount } =
counterSlice.actions

export default counterSlice.reducer
```

### Add Slice Reducers to the Store

Next, we need to import the reducer function from the counter slice and add it to our store. By defining a field inside the reducers parameter, we tell the store to use this slice reducer function to handle all updates to that state.

app/store.js

```
import { configureStore } from '@reduxjs/toolkit'
import counterReducer from '../features/counter/counterSlice'

export default configureStore({
  reducer: {
    counter: counterReducer,
  },
})
```

**Create Redux Folder & store file like belove**
**store  // create store and import on app.js**

```
import { createStore } from "redux";
import { reducer } from "./reducer";

const store = createStore(reducer);
export default store;
```

**type  // create type & import in reducer & action**

```
export const CHANGE_NAME = "CHANGE_NAME";
export const INCREMENT = "INCREMENT";
export const DECREMENT = "DECREMENT";
```

**action // create action & import in reducer**

```
import { CHANGE_NAME } from "./type";

export const changeName = () => {
  return {
    type: CHANGE_NAME,
  };
};
```

**reducer // create reducer and import in store**
**create state store in reducer**

```
import { act } from "react-dom/test-utils";

const initialState = {
    userName: "parth",
    number: 0,
  };

  export const reducer = (state = initialState, action) => {
    console.log(state, action);
    return state;
  }
```

Crud in react redux with react-toolkit

**npx create-react-app my-proj**
**npm install react-redux  // install redux for react**
**npm install @reduxjs/toolkit  // tool kit**

Lifecycle Hooks :  Used in Functional Component

Custom Component

---

```
// import liabrary for custome componant

import React,{Component} from 'react'; //   import React because react provide JSX code MEANS Javascript Xml FORMAT CODE for jsx format
import ReactDOM from 'react-dom';   // import ReactDOM for dom manuplation by componant


// page name & comp name are same and also Upercase fisrt letter
class Demo_comp extends Component
{
        // for call must rander function;
```

```
        render()
        {
                // must returm some value
                // its JSX code not HTML SO ITS VERTUAL dOM
                // only it reads one block so make one div must
                return <div>
                        <h1>Hello my cuustome ccomponant work </h1>
                        <p>hello ;)</p>
                </div>
        }
}


// after create and use componant must export componant
export default Demo_comp;

/*
after than just just add in index.js

=>import Demo_comp from './Demo_comp';  //  call custome componant name
and
=>ReactDOM.render(<Demo_comp />,document.getElementById('root'));  //
you can call app component from app.js
*/
```

---

**Demo.css**

**Also all component have their custom css also if you want**

**Create same component name Demo.css file & import in demo.js component**

**import './Demo_comp.css';**

**Now we all time use Class=”styleName” keywords but it is reserved keywords in javascript for create class so its work but not use as style.**

**So now for css or bootstrap react provide className instead of class keywords**

**If you want to use extra framework  like bootstrap / techyons**

**Techyons**

```
<link rel="stylesheet"
href="https://unpkg.com/tachyons@4.12.0/css/tachyons.min.css"/>
```

Also call link from this link
https://tachyons.io/docs/table-of-styles/

In react use

Cmd : **npm techyons**
For use : **import 'techyons';** in component

**---**

**Props** : means element attributes which used from main component too child component

Props get passed to the component
Props are immutable(cannot be changed/modify) only main component right so child not

Use in class component : **{this.props.name}**

# Function In react

Event most of used in function components

Just like HTML, React can perform actions based on user events.

React has the same events as HTML: click, change, mouseover etc.

Adding Events
React events are written in camelCase syntax:

onClick instead of onclick.

React event handlers are written inside curly braces:

onClick={shoot}  instead of onClick="shoot()".

<button onClick={shoot}>Take the Shot!</button>

you can also use arrow function withaout name of function  ()=>

const test= ()=>{

if you want to call direct bind than just use {test()}

}

*/

```
import React from 'react';   // MUST ADD FOR COMPONENT

function Event_func_component() {

  function test()
  {
        console.warn('functon called')
        alert('functon called');
  }

  const demo= ()=>{
        console.warn('its arrow function called')
        alert('its arrow function called');
  }


  return ( <div>
              <button onClick={test}>call simple function</button>
              <button onClick={demo}>call arrow function</button>
    </div>
```

```
  );
}
```

# What is ES6?

ES6 stands for ECMAScript 6.

ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015.

---

# Why Should I Learn ES6?

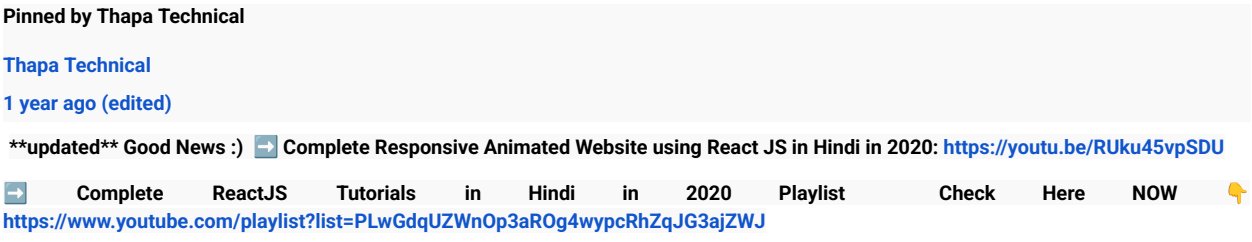
React uses ES6, and you should be familiar with some of the new features like:

- Classes
- Arrow Functions
- Variables (let, const, var)

# Classes

ES6 introduced classes.

A class is a type of function, but instead of using the keyword `function` to initiate it, we use the keyword `class`, and the properties are assigned inside a `constructor()` method.



Welcome, ReactJS Tutorial For Beginners in One Video in Hindi With One Mini-Project 2019. ReactJS basic from start with no step skipped in Hindi. 😊Good News,

➡️ Complete Responsive Animated Website using React JS in Hindi in 2020: https://youtu.be/RUku45vpSDU

➡️ Reactjs series start Kar Diya hai

Check Now 👍 Introduction to React | ReactJS Tutorial for Beginners in Hindi 2020 #1: https://youtu.be/tiLWCNFzThE

********** Must Watch Videos For Web Development **********

React tutorial for beginners in Hindi. Part 2 #ReactJS Routing Link: https://youtu.be/ikHh57o_JHU
ReactJS Part#3 Forms and validation in React: https://youtu.be/xbCHUq1VEUI

➡️ Hooks in React JS in Hindi | useState in Hook in React JS: https://youtu.be/SS1I7m-G2kk

➡️ Array Destructuring in ES6 in JavaScript in Hindi 2020: https://youtu.be/QANG6A1-zkQ

➡️ Object Destructuring in ES6 in JavaScript in Hindi: https://youtu.be/k6yMQhPvbrM

➡️ React DevTools & Source Code Link: https://www.thapatechnical.com/2020/0...

➡️ ReactJS Tutorial in Hindi 2020 Playlist Link: https://www.youtube.com/playlist?list...

➡️ ReactJS JavaScript Array Map Method in Hindi with Example: https://youtu.be/EETqnvQfpEg

➡️ Fat Arrow Function in JavaScript in Hindi 2020: https://youtu.be/y-boXLLwIHQ

➡️ #23: Array in React JS in Hindi | React JS Project Netflix App #2 in Hindi in 2020

➡️ Array in JavaScript in Hindi: https://youtu.be/zGlxOx_xSaY

➡️ #22: React JS Project Netflix App Part #1 in Hindi in 2020: https://youtu.be/HRhJVGjIraE

➡️ Install VS Code for ReactJS LINK: https://www.thapatechnical.com/2020/0...

➡️ Check  Complete Reactjs in One video here 👇 https://youtu.be/43IbFDSVdB0

➡️ HTML in One Video: https://youtu.be/pN7BFLfSjFw

➡️ CSS in One video:  https://youtu.be/hkQv0NIxKQY

➡️ CSS FlexBox in 30 Minutes: https://youtu.be/6L-PAmZajfA

➡️ JavaScript in One video:  https://youtu.be/PODgPW-hbfU

➡️ ECMAScript 6 in One Video:  https://youtu.be/Iek4DtRobLA

➡️ HTML5 in one video:  https://youtu.be/4V0-YpO-KZo

➡️ CSS3 in one video:  https://youtu.be/TFrFx7H5ZI8

➡️ Bootstrap4 in One video:  https://youtu.be/yZnNnvXO8HI

➡️ Jquery in One video: https://youtu.be/PNvyPEQ0y-I

➡️ JSON in one video: https://youtu.be/bMqevPKAPD4

➡️ ReactJS in one video: https://youtu.be/43IbFDSVdB0

➡️ PHP in One Video: https://youtu.be/3024Kc6SDJ0

➡️ NodeJS in one video: https://youtu.be/ipnWAKoiBt

➡️ MySQL in one video: https://youtu.be/5bFxbwjN-Gk

********** CLICK HERE TO WATCH **********

➡️ Fetch API in JavaScript: https://youtu.be/TojSXOuGUW4

➡️ AJAX tutorial for beginners in Hindi: https://www.youtube.com/playlist?list...

➡️ Template literals (Template strings) in ES6 in JavaScript in Hindi: https://youtu.be/Qcph3GB1JFs

➡️ Async Await in JavaScript in Hindi: https://youtu.be/hFLXE5-JCcs

➡️ **Promises in JavaScript in Hindi:** https://youtu.be/xGBhmi4wwMI

➡️ **Callback Hell in JavaScript:** https://youtu.be/fr67u98nckk

➡️ **Advanced JavaScript in Hindi Playlist:** https://www.youtube.com/playlist?list...

➡️ **ECMAScript Tutorial in Hindi 2020:** https://www.youtube.com/playlist?list...

➡️ **ES5 & ES6 | ECMAScript 6 in One Video in Hindi:** https://youtu.be/Iek4DtRobLA

➡️ **JavaScript Game Development Series in 2020:** https://www.youtube.com/playlist?list...