React

React is an open source javascript library for building user interfaces.

React not focussing on http requests & routing..

React is project created and maintained by the facebook.

React is component based archetexure. breakdown our application into small encapsulated parts which can then be composed more complex ui’s.

A traditional website can be broken down into HEADER, SIDENAV, MAIN CONETENT, FOOTER. Each section represents a component.

Component make it possible to write Re-usable codes.

React is declarative. i.e tell react what we want, react with its react DOM library built actual user interface.

React will make us painless to create complex ui’s by abstracting complex role form us.

React will handle efficiently in updating and rendering the components.

DOM updates are most expensive operations handled gracefully in react.

We can also enter the mobile applications with react.

IMPORTANCE OF EXPORT AND IMPORT

/React/happie-world/src/App.css

//this is Greet.js

import React from 'react'

 const Greet = ()=><h1>architecture</h1>

export default Greet //this export default allows the Greet.js file with anyname

//this is App.js

import MyComponent from './components/Greet'

import React, { Component } from 'react';

class App extends Component{

  render(){

  return(

    <div className = "App">

      <MyComponent/>

    </div>

  );

}

}

export default App; //exporting the Greet.js as another name MyComponent

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//This is Greet.js

import React from 'react'

export const Greet = ()=> <h1>don't hangout with the people who take out value from your life </h1> //this is named export..definitely we have to export with the excat name..

//this App.js

import { Greet } from './components/Greet';

import { Component } from 'react';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Greet/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Greet.js

import React from 'react'

function Greet(){

    return <h1>hello karthik yadav</h1>

}

export default Greet

//App.js

import logo from './logo.svg';

import './App.css';

import Greet from './components/Greet';

function App() {

  return (

    <div className="App">

      <Greet/>

    </div>

  );

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Greet.js

import React from 'react'

const Greet = ()=> <h1>jai hanuman</h1> //arrow functions

export default Greet

//App.js

import './App.css';

import Greet from './components/Greet';

function App() {

  return (

    <div className="App">

      <Greet/>

    </div>

  );

}

export default App;

//props................................................................

props are arguments passed in react components....

props can be passed to react componets through the HTML attributes..

react props are the arguments in the functions..attributes in the HTML..

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

functional components

//this is Greet.js………………………………………………….

import React from 'react'

const Greet = ()=> <h1>jai hanuman</h1>

export default Greet

//this is App.js…………………………………………………………..

import MyComponent from './components/Greet' //exporting the Greet.js name

import React, { Component } from 'react' with other name like

import './App.css'; myComponent

class App extends Component{

  render(){

  return(

    <div className = "App">

      <MyComponent/>

    </div>

  );

}

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Greet.js

import React from 'react'

 const Greet = ()=><h1>archetecture</h1>

export default Greet

//this is App.js

import React from 'react'

import Greet from './components/Greet';

import './App.css';

function App(){

  return(

    <div className = "App">

      <Greet/>

    </div>

  )

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Greet.js

import React from 'react'

function Greet(){

    return <body bgcolor ="aqua">

        <h1>krishna yadav</h1>

        <h2>karthik yadav</h2>

    </body>

}

export default Greet

//this is App.js

import React from 'react'

import Greet from './components/Greet'

import './App.css';

function App(){

  return(

    <div className = "App">

      <Greet/>

    </div>

  )

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Hello.js

import React from 'react'

 const Hello = function(){

     return(

         <h1>this is an anonymous function</h1>

     )

 }

 export default Hello

//this is App.js

import React from 'react'

import Hello from './components/Hello'

import './App.css';

function App(){

  return(

    <div className = "App">

      <Hello/>

    </div>

  )

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class components

//this is Welcome.js

import React, {Component} from 'react'

class Welcome extends Component {

    render(){

        return <h1>courtesy is the short distance between the two people</h1>

    }

}

export default Welcome;

//this is App.js

import React from 'react'

import Welcome from './components/Welcome'

import './App.css';

function App(){

  return(

    <div className = "App">

      <Welcome/>

    </div>

  )

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Welcome.js

import React, {Component} from 'react'

class Welcome extends Component {

    render(){

        return <h1>courtesy is the short distance between the two people</h1>

    }

}

export default Welcome;

//this App.js

import MyComponent from './components/Welcome'

import React, { Component } from 'react'

import './App.css';

class App extends Component{

  render(){

  return(

    <div className = "App">

      <MyComponent/>

    </div>

  );

}

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Props with functional components

//Greet.js is the functional Component

import React from 'react'

const Greet = (props)=>{

    console.log(props);

    return(

        <div>

        <h1>

            hello {props.name} & {props.native}

        </h1>

        {props.children}

        </div>

    )

}

export default Greet;

//App.js

import React, {Component} from 'react';

import './App.css';

import Greet from './components/Greet';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Greet name = "Rama koti yadav" native = "rani peta">

          <p>this is children</p>

        </Greet>

        <Greet name = "karthik yadav" native = "godalam">

          <p>son</p>

        </Greet>

        <Greet name = "krishna yadav" native = "godalam">

          <p>father</p>

        </Greet>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Greet.js which is functional component………….

import React from 'react'

const Greet = (props)=>{

    console.log(props);

    return(

        <div>

        <h1>MyByk:{props.myByk}</h1>

        </div>

    )

}

export default Greet;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Greet.js...............

import React from 'react'

const Greet = (props)=>{

    console.log(props);

    return<h1>hello {props.Name} & {props.hus} </h1>

}

export default Greet

//this is App.js........................

import React, {Component} from 'react';

import './App.css';

import Greet from './components/Greet';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Greet Name = "gajalaxmi paramaivam" hus = "karthik yadav"/>

        <Greet Name = "sunitha ankaboina" hus = "laxmi narayana"/>

        <Greet Name = "anitha korlana" hus = "manoj"/>

      </div>

    );

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this Greet.js......................

import React from 'react'

const Greet = (props)=>{

    console.log(props);

    return(

        <div>

        <h1>

            hello {props.name} & {props.native}

        </h1>

        {props.children}

        </div>

    )

}

export default Greet;

//this App.js................

import React, {Component} from 'react';

import './App.css';

import Greet from './components/Greet';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Greet name = "Rama koti yadav" native = "rani peta">

          <p>this is children</p>

        </Greet>

        <Greet name = "karthik yadav" native = "godalam">

          <p>son</p>

        </Greet>

        <Greet name = "krishna yadav" native = "godalam">

          <p>father</p>

        </Greet>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this Greet.js…………………………………………………………………………….

import React from 'react'

const Greet = function(props){

     console.log(props);

     return(

         <div>

             <h1>

                 mySelf{props.name}

             </h1>

             {props.children}

         </div>

     )

}

export default Greet;

//this App.js……………………………………………………………….

import React,{Component} from 'react'

import './App.css';

import Greet from './components/Greet';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Greet name = "karthik yadav">

          <p>MSc botany andhraUniversity</p>

        </Greet>

      </div>

    )

  }

}

export default  App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Greet.js…………………………………………………………

import React from 'react'

const Greet=(props)=>{

    console.log(props)

    return <h1>hello karthik yadav</h1>

}

export default Greet

//this App.js…………………………………………………………………………

import React from 'react';

import Greet from './components/Greet';

function App(){

  return(

    <div className = "APP">

          <Greet/>

    </div>

  )

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//passing...................Props are also how you pass data from one component to another, as parameters.

//this is Hello.js

import React from 'react';

const Hello = (props)=>{

    console.log(props);

    return(

        <h1>mySelf{props.name}</h1>

    )

}

export default Hello;

//this is App.js…………………………………………………..

import React,{Component} from 'react'

import Hello from './components/Hello'

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <h1>MSc Botany andhraUniversity</h1>

        <Hello name = "karthik yadav"/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this Greet.js

import React from 'react'

const Hello = (props)=>{

        console.log(props);

        return(

            <h1>inflorescence {props.specialInflo}</h1>

        )

}

export default Hello;

//this App.js

import React, {Component} from 'react'

import Hello from './components/Hello'

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Hello specialInflo = "cyathium vertticellaster"/>

        <h1>special type of inflorescence</h1>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//another way of assigning the variable to const...................

//Hello.js

import React from 'react'

const Hello = (props)=>{

        console.log(props);

        return(

            <h1>inflorescence {props.specialInflo}</h1>

        )

}

export default Hello;

//App.js

import React, {Component} from 'react'

import Hello from './components/Hello'

import './App.css';

class App extends Component{

  render(){

    const flower = "cyathium vertecellaster"

    return(

      <div className = "App">

        <Hello specialInflo = {flower}/>

        <h1>special type of inflorescence</h1>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Props with class components

//this is Welcome.js………………………………………….

import React, {Component} from 'react'

class Welcome extends Component {

    render(){

        return <h1>courtesy is the short distance between the two people</h1>

    }

}

export default Welcome;

//App.js……………………………………………………………….

import React from 'react'

import './App.css';

import Welcome from './components/Welcome';

function App() {

  return (

    <div className="App">

      <Welcome/>

    </div>

  );

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is importing the two components like fun and class into App.js

//Greet.js………………………………………

import React from 'react'

function Greet(){

    return <h1>hello karthik yadav</h1>

}

export default Greet;

//Welcome.js…………………………………………….

import React, {Component} from 'react'

class Welcome extends Component {

    render(){

        return <h1>courtesy is the short distance between the two people</h1>

    }

}

export default Welcome;

//App.js…………………………………………………..

import React from 'react'

import Greet from './components/Greet';

import Welcome from './components/Welcome';

import './App.css';

function App(){

  return(

    <div className = "App">

      <Greet/>

      <Welcome/>

    </div>

  )

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

notice:-this is the class component….

//Hello.js…..

import React, {Component} from 'react'

 class Hello extends Component {

     render(){

         return <h1>MyScooty:{this.props.Scooty}</h1>

     }

 }

export default Hello;

//App.js

import React, { Component } from 'react';

import './App.css';

import Hello from './components/Hello';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <Hello Scooty = "Suzuki Access"/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Welcome.js

import React, {Component} from 'react'

class Welcome extends Component {

    render(){

        return <h1>cucurbitaceae family:{this.props.name}</h1>

    }

}

export default Welcome;

//this is App.js

import React, { Component } from 'react';

import './App.css';

import Welcome from './components/Welcome';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <Welcome name = "mimordica charantica"/>

        <Welcome name  = "luffa aegyptica"/>

        <Welcome name = "luffa accutangula"/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

State in Class Components

//this is Message.js //1st step is to Create State object &

import React, {Component} from 'react' initialise

class Message extends Component{ //2nd step bind this.state value

    constructor(){ to render().

        super() //3rd step An HTML for a button element

        this.state = { //4th step listen to onClick event to this

     message:'welcome to web designing tutorials' button & changeMessage()

        }

    }

    changeMessage(){

        this.setState({

            message: 'thanks for watching'

        })

    }

    render(){

        return( //return() parenthesis require JSX expands our multipleLines

            <div> //inclusive <div> tag

            <h1>{this.state.message}</h1>

            <button onClick = {()=> this.changeMessage()}>subscribe</button>

            </div>

        )

    }

}

export default  Message

//this is App.js

import { Component } from 'react';

import Message from './components/Message';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Message/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

setState() in class Component

setState()

//always make use of setState and never modify the state directly

//if the certain code has been executed after the state has been updated?place the code in the callback function which is the 2nd parameter to the setState()

//when you have to update the state based on the previous state value.pass in a function as an argument instead of the regular object

//this is Counter.js

import React,{Component} from 'react'

class Counter extends Component{

    constructor(props){

        super(props)

        this.state = {

            count:0

        }

    }

    Increment(){

        this.state.count = this.state.count + 1

        console.log(this.state.count)

    }

    render(){

        return(

        <div>

            <div>count-{this.state.count}</div>

            <button onClick = {()=> this.Increment()}>Increment</button>

        </div>

        ) //UI is not Re-rendering

    }

}

export default Counter

//this is App.js

import React, { Component } from 'react'

import Counter from './components/Counter';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Counter/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

////this is Counter.js

import React,{Component} from 'react'

class Counter extends Component{

    constructor(props){

        super(props)

        this.state = {

            count:0

        }          //never modify rthe state the directly

    }              // instaed make use of setState.

    Increment(){

        this.setState({ //calls to setState() are asynchronous

            count:this.state.count+1

        })

        console.log(this.state.count)

    }

    render(){

        return(

        <div>

            <div>count-{this.state.count}</div>

            <button onClick = {()=> this.Increment()}>Increment</button>

        </div>

        )

    }

}

export default Counter

//this is App.js

import React, { Component } from 'react'

import Counter from './components/Counter';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Counter/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Counter.js

import React,{Component} from 'react'

class Counter extends Component{

    constructor(props){

        super(props)

        this.state = {

            count:0

        }

    }

    Increment(){

        this.setState(

            {    //setState() has two parameters 1)setState Object 2)Callback Arrow function

            count:this.state.count+1

        },

        ()=>{

            console.log('Callback value',this.state.count)   //1 is from Callback function console.log statement

        })

        console.log(this.state.count)  //0 is from Synchronous console.log

    }

    render(){

        return(

        <div>

            <div>count-{this.state.count}</div>

            <button onClick = {()=> this.Increment()}>Increment</button>

        </div> //whenever you need to execute some code after the state has

        ) been changed do place the code right after setState().instead

    } palce that code within the callback function that is passed as 2nd

} parameter to the setState()

export default Counter

//this is App.js

import React, { Component } from 'react'

import Counter from './components/Counter';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Counter/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this Counter.js

import React,{Component} from 'react'

class Counter extends Component{

    constructor(props){

        super(props)

        this.state = {

            count:0

        }

    }

    Increment(){

        this.setState(

            {    //setState() has two parameters 1)setState Object 2)Callback Arrow function

            count:this.state.count+1

        },

        ()=>{

            console.log('Callback value',this.state.count)   //1 is from Callback function console.log statement

        })

        console.log(this.state.count)  //0 is from Synchronous console.log

    }

    IncrementFive(){

        this.Increment()   //0 locked 5 times

        this.Increment()   //1 locked 5 times from Callback

        this.Increment()  //this behaviour of React may grouped multiple seState calls

        this.Increment()  //into a single update fro better performence all the 5 setState calls

        this.Increment()  //are done in dene in one single go updated value doesnot carry over b/w

    }      //the different calls.so whenever you have  updated the state based on the previous state we need pass

    render(){ //function as an argument to setstate() instead of passing in an object

        return(

        <div>

            <div>count-{this.state.count}</div>

            <button onClick = {()=> this.IncrementFive()}>Increment</button>

        </div>

        )

    }

}

export default Counter

//this is App.js

import React, { Component } from 'react'

import Counter from './components/Counter';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Counter/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Counter.js

import React,{Component} from 'react'

class Counter extends Component{

    constructor(props){

        super(props)

        this.state = {

            count:0

        }

    }

    Increment(){

        /\*this.setState(

            {    //setState() has two parameters 1)setState Object 2)Callback Arrow function

            count:this.state.count+1

        },

        ()=>{

            console.log('Callback value',this.state.count)   //1 is from Callback function console.log statement

        })

        console.log(this.state.count)  //0 is from Synchronous console.log\*/

        this.setState((prevState)=>({

            count:prevState.count+1   //when we update the state based on previous state make sure passing the function as an argument instaed of regular object

        }))

        console.log(this.state.count)

    }

    IncrementFive(){

        this.Increment()   //0 locked 5 times

        this.Increment()   //1 locked 5 times from Callback

        this.Increment()  //this behaviour of React may grouped multiple seState calls

        this.Increment()  //into a single update fro better performence all the 5 setState calls

        this.Increment()  //are done in dene in one single go updated value doesnot carry over b/w

    }      //the different calls.so whenever you have  updated the state based on the previous state we need pass

    render(){ //function as an argument to setstate() instead of passing in an object

        return(

        <div>

            <div>count-{this.state.count}</div>

            <button onClick = {()=> this.IncrementFive()}>Increment</button>

        </div>

        )

    }

}

export default Counter

//this is App.js

import React, { Component } from 'react'

import Counter from './components/Counter';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Counter/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DE-structuring props in functional components

destructuring in the parameter

//this is Greet.js

import React from 'react'

const Greet = ({Name,hus})=>{

    return(

        <div>

    <h1>hello {Name} & {hus} </h1>

    </div>

    )

}

export default Greet

//this is App.js

import React, {Component} from 'react'

import Greet from './components/Greet';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Greet Name = "gajalaxmi paramaivam" hus = "karthik yadav"/>

      </div>

    );

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Greet.js

destructuring in the function body

import React from 'react'

const Greet = (props)=>{

    const {Name, hus} = props

    return(

        <div>

    <h1>hello {Name} & {hus} </h1>

    </div>

    )

}

export default Greet

//this is App.js

import React, {Component} from 'react'

import Greet from './components/Greet';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Greet Name = "gajalaxmi paramaivam" hus = "karthik yadav"/>

      </div>

    );

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Destructuring the props in class Components

in render()

//this is Welcome.js

import React, {Component} from 'react'

class Welcome extends Component {

    render(){

        const {name} = this.props

        return <h1>cucurbitaceae family:{name}</h1>

    }

}

export default Welcome;

//this is App.js

import React, { Component } from 'react';

import './App.css';

import Welcome from './components/Welcome';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <Welcome name = "mimordica charantica"/>

        <Welcome name  = "luffa aegyptica"/>

        <Welcome name = "luffa accutangula"/>

      </div>

    )

  }

}

export default App;

EventHandling in functional Components

//this is FunctionClick

import React from 'react'

function FunctionClick(){ //with jsx you pass  function as the eventHandler rather than a string

    function clickHandler(){

        console.log('Button clicked');

    }

    return(   //React Events are named using camelCase

        <div>

        <button  onClick = {clickHandler}>click</button>

        </div>

    ) //here clickHandler is the function..

}

export default FunctionClick

//this is App.js

import React from 'react'

import FunctionClick from './components/FunctionClick';

import './App.css';

function App(){

  return(

    <div className = "App">

      <FunctionClick/>

    </div>

  )

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

EventHandlers must be function not function call

//this is FunctionClick.js

import React from 'react'

function FunctionClick(){

    function clickHandler(){ //we want the EventHandler to be function not as function call

        console.log('Button clicked');

    }

    return(  //clickHandler() is the function call

        <div>

        <button  onClick = {clickHandler()}>click</button>

        </div>

    )

}

export default FunctionClick

//this is App.js

import React from 'react'

import FunctionClick from './components/FunctionClick';

import './App.css';

function App(){

  return(

    <div className = "App">

      <FunctionClick/>

    </div>

  )

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

eventHandling in class Components

//this is ClassClick.js

import React,{Component} from 'react'

class ClassClick extends Component{

    clickHandler(){

        console.log('clicked the button');

    }

    render(){

        return(//in class Component methods can be accessed through this keyword

            <div>

               <button onClick = {this.clickHandler}>click me</button>

            </div>

        )

    }

}

export default ClassClick

//this is App.js

import React,{Component} from'react'

import ClassClick from './components/ClassClick';

import './App.css'

class App extends Component{

  render(){

    return(

      <div className = "App">

        <ClassClick/>

      </div>

    )

  }

}

export default App;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Binding events handler

//this is EventBind.js

Approch one binding in render method

import React,{Component} from 'react'

class EventBind extends Component{

     constructor(){

         super()

         this.state = {

             message:'Hello'

         }

     }

     clickHandler(){

         this.setState({

             message:"Good Bye"

         })

         console.log(this)   //this keyword refers event bind component

     }

    render(){

        return(

            <div>

                <div>{this.state.message}</div>

                 <button onClick = {this.clickHandler.bind(this)}>click</button>

            </div>

        )

    }

}

export default EventBind;

//this is App.js

import React,{Component} from 'react';

import EventBind from './components/EventBind';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <EventBind/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

arrow functions in render method

//this is EventBind.js

import React,{Component} from 'react'

class EventBind extends Component{

     constructor(){

         super()

         this.state = {

             message:'Hello'

         }

     }

     clickHandler(){

         this.setState({

             message:"Good Bye"

         })

         console.log(this)   //this keyword refers event bind component

     }

    render(){

        return(//we don't need the curly braces and return keyword for()=> because the function body is a single statement

            <div>

                <div>{this.state.message}</div>

                 <button onClick = {()=>this.clickHandler()}>click</button>

            </div>

        ) //we are calling the eventHandler are returning the value that is why paranthesis is required in this approch

    }

}

export default EventBind;

//this is App.js

import React,{Component} from 'react';

import EventBind from './components/EventBind';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <EventBind/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is EventBind.js

Approch 3,this is binding the eventhandler in the constructor

import React,{Component} from 'react'

class EventBind extends Component{

     constructor(props){

         super(props)

         this.state = {

             message:'Hello'

         }

         this.clickHandler = this.clickHandler.bind(this) //this method is

     }

     clickHandler(){ mostly used in React documentation

         this.setState({

             message:"Good Bye"

         })

         console.log(this)

     }

    render(){

        return(

            <div>

                <div>{this.state.message}</div>

                 <button onClick = {this.clickHandler}>click</button>

            </div>

        )

    }

}

export default EventBind;

//this is App.js

import React,{Component} from 'react';

import EventBind from './components/EventBind';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <EventBind/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is EventBind.js

arrow function as class property

import React,{Component} from 'react'

class EventBind extends Component{

     constructor(props){

         super(props)

         this.state = {

             message:'Hello'

         }

     }

     clickHandler = ()=>{

         this.setState({

             message:'Good Bye!'

         })

     }

    render(){

        return(

            <div>

                <div>{this.state.message}</div>

                 <button onClick = {this.clickHandler}>click</button>

            </div>

        )

    }

}

export default EventBind;

//this is App.js

import React,{Component} from 'react';

import EventBind from './components/EventBind';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <EventBind/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is ParentComponent.js

methods as props

import React,{Component} from 'react';

import ChildComponent from './ChildComponent';

class ParentComponent extends Component{

    constructor(){

        super()

        this.state = {

          parentName:'krishna yadav'

        }

        this.greetParent = this.greetParent.bind(this)

    }

    greetParent(){

        //alert('Hello'+this.state.parentName)

        alert(`Hello ${this.state.parentName}`)backticks are feature of ES6 no

    } t react js

    render(){

        return(

            <div>

                <ChildComponent greetHandler = {this.greetParent}/>

            </div>

        )

    }

}

export default ParentComponent

//this is ChildComponent.js

import React from 'react'

function ChildComponent(props){

    return(   passing the methods as the props to child component

        <div>

        <button onClick = {props.greetHandler}>greetParent</button>

        </div>

    )

}

export default ChildComponent

//this is App.js

import React,{Component} from 'react';

import ParentComponent from './components/ParentComponent';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <ParentComponent/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is ParentComponent.js

import React,{Component} from 'react';

import ChildComponent from './ChildComponent';

class ParentComponent extends Component{

    constructor(){

        super()

        this.state = {

          parentName:'parent'

        }

        this.greetParent = this.greetParent.bind(this)

    }

    greetParent(childName){

        //alert('Hello'+this.state.parentName)

        alert(`Hello ${this.state.parentName} from ${childName}`)  //backticks are feature of ES6 not react js

    }

    render(){

        return(

            <div>

                <ChildComponent greetHandler = {this.greetParent}/>

            </div>

        )

    }

}

export default ParentComponent

//this is ChildComponent.js

passing the parameters when calling the parent method  from the child component..in this way ()=> in return statement is useful

()=> is simplest way to pass the parameters form the childcomponents to parentComponent

import React from 'react'

function ChildComponent(props){

    return(  //passing the methods as the props to child component

        <div>

        <button onClick = {()=>props.greetHandler('child')}>greetParent</button>

        </div>

    )

}

export default ChildComponent

//this is App.js

import React,{Component} from 'react';

import ParentComponent from './components/ParentComponent';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <ParentComponent/>

      </div>

    )

  }

}

export default App

in the ParentComponent define the method and ChildComponent pass the method as the prop.in the ChildComponent access the method using props object.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Conditional rendering

if else statements don't work inside the JSX because jsx is just syntactic sugar for function calls and obj construction

adding if else statementswith in JSX ids not valid.entaire return statements conatined the JSX statements displayed inside

if or else block

//this is UserGreet.js

import React,{Component} from'react'

class UserGreeting extends Component{

    constructor(){

        super()

        this.state = {

            isLoggedIn:false

            //isLoggedIn:true

        }

    }

    render(){

        if(this.state.isLoggedIn){

            return(

                <div>Hello karthik yadav</div>

            )

        }else{

            return(

                <div>hello guest</div>

            )

        }

    }

}

export default UserGreeting

//this is App.js

import React,{Component} from 'react';

import UserGreeting from './components/UserGreeting';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <UserGreeting/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is UserGreeting.js

element variables...........

in this approch we use javaScript variables  to store elements this will also help conditonally render the entaire component and the part of component aswell

import React,{Component} from'react'

class UserGreeting extends Component{

    constructor(){

        super()

        this.state = {

            //isLoggedIn:false

            isLoggedIn:true

        }

    }

    render(){

        let message

        if(this.state.isLoggedIn){

            message = <div>Hello karthik yadav</div>

        }else{

            message = <div>Hello guest</div>

        }

        return <div>{message}</div>

    }

}

export default UserGreeting

//this is App.js

import React,{Component} from 'react';

import UserGreeting from './components/UserGreeting';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <UserGreeting/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is UserGreeting.js

ternary conditinal operator..

it can be used inside the JSX

import React,{Component} from'react'

class UserGreeting extends Component{

    constructor(){  //we follow this method most of the times keeps code simple and readable

        super()

        this.state = {

            //isLoggedIn:false

            isLoggedIn:true

        }

    }

    render(){

        return this.state.isLoggedIn ? (

            <div>Hello karthik yadav</div>

        ) : (

           <div> Hello guest</div>

        )

    }

}

export default UserGreeting

//this is App.js

import React,{Component} from 'react';

import UserGreeting from './components/UserGreeting';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <UserGreeting/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is UserGreeting.js

Short circuit operator method approch....

this is specific case of tenary method..when you want render something or nothing.. you make use of Short circuit operator method approch

import React,{Component} from'react'

class UserGreeting extends Component{

    constructor(){

        super()

        this.state = {

            isLoggedIn:false

            //isLoggedIn:true

        }

    }

    render(){

        return this.state.isLoggedIn && <div>Hello karthik yadav</div>

    }

}

export default UserGreeting

//this is App.js

import React,{Component} from 'react';

import UserGreeting from './components/UserGreeting';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <UserGreeting/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

List rendering

//This is NameList.js

import React from 'react'

function NameList(){

    const names = ['karthik','Rajesh','Joga','Girish']

    return(

        <div>

            <h2>{name[0]}</h2>

            <h2>{name[1]}</h2>

            <h2>{name[2]}</h2>

            <h2>{name[3]}</h2>

        </div>

    )

}

export default NameList

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is NameList.js

import React from 'react'

function southStates(){

    const states = ["Andhra pradhesh","Telangana","TamilNadu","kerala","karnataka"]

    return( //function name is changed as per my wish not only like

        <div> file name(NameList.js)

            <h1>Southern states in india</h1>

            <h1>{states[0]}</h1>

            <h1>{states[1]}</h1>

            <h1>{states[2]}</h1>

            <h1>{states[3]}</h1>

            <h1>{states[4]}</h1>

        </div>

    )

}

export default southStates

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is NameList.js

import React from 'react'

function NameList(){

    const names = ['karthik','Rajesh','Joga','Girish']

    return(

        <div>

        {

         names.map((name) => <h2>{name}</h2>)

        }

        </div>

    )

}

export default NameList

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is NameList.js

import React from 'react'

function NameList(){

    const names = ['karthik','Rajesh','Joga','Girish']

    const nameList = names.map(name => <h2>{name}</h2>)

    return(

        <div>{nameList}</div>

    )

}

export default NameList

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is NameList.js

import React from 'react'

function NameList(){

    const data = [

    {

       id:1,

       name:"karthik yadav",

       age:"26",

       skill:"User Interphase"

    },

    {

        id:2,

        name:"Rajesh",

        age:"27",

        skill:"botanist"

    },

    {

        id:3,

       name:"Joga",

       age:"27",

       skill:"Teacher"

    },

    {

        id:4,

       name:"Girish",

       age:"26",

       skill:"very Talkative"

    }            //recomonded JSX into seperate component

]

    const dataOfFriends = data.map(person =>(

         <h2>

             MySelf {person.name}. i am {person.age}year old. i am {person.skill}

             </h2>

        ))

    return(

        <div>{dataOfFriends}</div>

    )

}

export default NameList

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is NameList.js

import React from 'react'

import Person from './Person'

function NameList(){

    const persons = [

    {

       id:1,

       name:"karthik yadav",

       age:"26",

       skill:"User Interphase"

    },

    {

        id:2,

        name:"Rajesh",

        age:"27",

        skill:"botanist"

    },

    {

        id:3,

       name:"Joga",

       age:"27",

       skill:"botany faculty"

    },

    {

        id:4,

       name:"Girish",

       age:"26",

       skill:"botany faculty"

    }            //recomonded Re-Factor JSX into seperate component..

]

    const personList = persons.map(person =>  <Person person  = {person}/>)

    return <div>{personList}</div>

}

export default NameList

//this is Person.js

import React from 'react'

function Person({person}){

    return(

        <div>

            <h2>

         MySelf{person.name}.i am {person.age}years old. i know{person.skill}

        </h2>

        </div>

    )

}

export default Person

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lists and keys

In the above code When we render the list of items React throws a warning in console ‘Each child in an array or iterator should have a unique “key” prop.

In simpler words React telling us, hey each item in the list render using map operator should have a prop called key and the value to the prop should be unique with in the list. Key prop value need not id all the time.it could even

Be a name.it really can be anything as long as u know for sure unique within the list. Key props is the special attribute you need include in creating lists of elements.

//this is NameList.js

import React from 'react'

import Person from './Person'

function NameList(){

    const persons = [

        {

           id:1,

           name:"karthik yadav",

           age:"26",

           skill:"User Interphase"

        },

        {

            id:2,

            name:"Rajesh",

            age:"27",

            skill:"botanist"

        },

        {

            id:3,

           name:"Joga",

           age:"27",

           skill:"botany faculty"

        },

        {

            id:4,

           name:"Girish",

           age:"26",

           skill:"botany faculty"

        }            //recomonded Re-Factor JSX into seperate component..

    ]

        const personList = persons.map(person =>  <Person key = {person.name} person  = {person}/>)

     //key props are not accessable in child component

return <div>{personList}</div>

}

export default NameList

//this is Person.js

import React from 'react'

function Person({person}){    // react say key props is reserved and something i need to render the list efficiently if you are try to pass the value in the child component pass it as different prop.

    return(

        <div>

            <h2>

         MySelf {person.name}.i am {person.age} years old. i know {person.skill}

        </h2>

        </div>

    )

}

export default Person

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Key props are not accessible in child component

//this is NameList.js

import React from 'react'

import Person from './Person'

function NameList(){

    const persons = [

        {

           id:1,

           name:"karthik yadav",

           age:"26",

           skill:"User Interphase"

        },

        {

            id:2,

            name:"Rajesh",

            age:"27",

            skill:"botanist"

        },

        {

            id:3,

           name:"Joga",

           age:"27",

           skill:"botany faculty"

        },

        {

            id:4,

           name:"Girish",

           age:"26",

           skill:"botany faculty"

        }            //recomonded Re-Factor JSX into seperate component..

    ]

        const personList = persons.map(person =>  <Person key = {person.name} person  = {person}/>)

     //key props are not accessable in child component

return <div>{personList}</div>

}

export default NameList

//this is Person.js

import React from 'react'

function Person({person,key}){    // react say key props is reserved and something i need to render the list efficiently if you are try to pass the value in the child component pass it as different prop.

    return(

        <div>

            <h2>

         {key}MySelf {person.name}.i am {person.age} years old. i know {person.skill}

        </h2>

        </div>

    )

}

export default Person

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Keys helps React identifying which items In list changed or added or removed

Plays a crucial role in handling the ui update efficiently

List without Key Attribute

<ul> <ul>

<li>karthik yadav</li>------------> <li>karthik yadav</li>

<li>krishna yadav</li>------------> <li>krishna yadav</li> <li>Sivaji yadav</li>-------------> Mutaion

</ul> </ul>

<ul> <ul>

<li>karthik yadav</li>--------------> mutation <li>karthik yadav</li>

<li>krishna yadav</li>--------------> mutation <li>karthik yadav</li> <li>Sivaji yadav</li>---------------> mutation

</ul> </ul>

In order to solve this problem react support key Attribute.

List with Key Attribute

<ul> <ul>

<li key=”1”>karthik yadav</li> <li key=”1”>Sivaji yadav</li>

<li key=”1”>krishna yadav</li> <li key=”1”>karthik yadav</li> <li key=”1”>Sivaji yadav</li> <li key=”1”>Krishna yadav</li>

</ul> </ul>

When the key is with the children component in the tree.it simply matches children with key the subsequent tree.and new children will simply add on the top of the list..

@ A “key” is a special string attribute you need to include when creating lists of elements.

@ Keys gives the elements a stable identity.

@ Keys helps React identity which items have changed or added or removed

@ This results in much efficient update of the user interface.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

index as key in rendering the list

//this is NameList.js

import React from 'react'

function NameList(){

    const names = ["karthik yadav","Rajesh","Joga","Girish"]

  const nameList = names.map(name =>  <h2>{name}</h2>)

return <div>{nameList}</div>

}

export default NameList

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

each child in the array or iterator should have a unique "key" prop.

we don't have an id that uniquely idetifies item in the array.we could use name itself as the key.

//this is NameList.js

import React from 'react'

import Person from './Person'

function NameList(){

    const names = ["karthik yadav","Rajesh","Joga","Girish"]

  const nameList = names.map(name =>  <h2 key = {name}>{name}</h2>)

return <div>{nameList}</div>

}

export default NameList

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Duplication of the names..but Key prop has to be unique.

Encountered two children with the same key, `Joga`.

Keys should be unique so that components maintain their identity across updates.

 Non-unique keys may cause children to be duplicated and/or omitted —

 the behavior is unsupported and could change in a future version.

//this is NameList.js

import React from 'react'

import Person from './Person'

function NameList(){

    const names = ["karthik yadav","Rajesh","Joga","Girish","Joga"]

  const nameList = names.map((name) =>  <h2 key = {name}>{name}</h2>)

return <div>{nameList}</div>

}

export default NameList

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

to fix above problem we have to apply index value as key for list rendering..every item recieves a unique key value..

using index as key causing some serious UI issues

when to use index as key.

1. The items in your list do not have a unique id.

2. The list is a static and will not change. Eg: never adding or removing items

To list.

3. The list will never be reordered or filtered. Eg:sorting based on a column value and filtering based on user inputs.

//this is NameList.js

import React from 'react'

function NameList(){

    const names = ["karthik yadav","Rajesh","Joga","Girish","Joga"]

  const nameList = names.map((name,index) =>  <h2 key = {index}>{index} {name}</h2>)

return <div>{nameList}</div>

}

export default NameList

//this is App.js

import React,{Component} from 'react';

import NameList from './components/NameList';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <NameList/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Styling and CSS Basics

//this is Stylesheet.js

import React from 'react'

import './myStyles.css';

function Stylesheet(){

    return(

        <div>

            <h1 className = "primary">Karthik yadav</h1>

        </div>

    )

}

export default Stylesheet

//this is myStyles.css

    .primary {

    color:darkorange

}

//this is App.js

import React,{Component} from 'react';

import Stylesheet from './components/Stylesheet';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

      <Stylesheet/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Stylesheet.js

import React from 'react'

import './myStyles.css';

function Stylesheet(props){

    let className = props.primary ? 'primary' : ''

    return(

        <div>

            <h1 className = {className}>Karthik yadav</h1>

        </div>

    )

}

export default Stylesheet

//this is myStyles.css

.primary {

    color:darkorange

}

//This is App.js

import React,{Component} from 'react';

import Stylesheet from './components/Stylesheet';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

      <Stylesheet primary = {false}/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Stylesheet.js

import React from 'react'

import './myStyles.css';

function Stylesheet(props){

    let className = props.primary ? 'primary' : ''

    return(

        <div>

            <h1 className = {className}>Karthik yadav</h1>

        </div>

    )

}

export default Stylesheet

//this is myStyles.css

.primary {

    color:darkorange

}

//this is App.js

import React,{Component} from 'react';

import Stylesheet from './components/Stylesheet';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

      <Stylesheet primary = {true}/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If we want to specify multiply classes

//if we want multiple classes the simplest way is to use template literals

// this is Stylesheet.js

import React from 'react'

import './myStyles.css';

function Stylesheet(props){

    let className = props.primary ? 'primary' : ''

    return(

        <div>

            <h1 className = {`${className}.font-xl`}>Tagetus patula</h1>

        </div>

    )

}

export default Stylesheet

//this is Inline.js

//created object applied to stylesheet

import React from 'react';

const heading = {

    fontSize:'72px',

    color:"blue"

}

function Inline(){

    return(

        <div>

            <h1 style = {heading}>Inline</h1>

        </div>

    )

}

export default Inline

//this is App.js

import React,{Component} from 'react';

import Stylesheet from './components/Stylesheet';

import Inline from './components/Inline';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <Inline/>

      <Stylesheet primary = {true}/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

this is regular style sheet.....

//this is appStyle.css

.error {

    color:coral

}

//this is appStyles.module.css

This is regular CSS StyleSheet

.success{

    color:cyan

}

//this is App.js

import React,{Component} from 'react';

import  styles from './appStyles.module.css';

import './appStyles.css';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <h1 className = 'error'>Error</h1>

        <h1 className = {styles.success}>Success</h1>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Form Handling

//this is Form.js

import React,{ Component} from 'react'

import './myStyles.css';

class Form extends Component{

    render(){

        return(

            <div className = "primary">

            Form Handling

            </div>

        )

    }

}

export default Form

//this is myStyles.css

.primary {

    color:olive;

}

//this is App.js

import React,{Component} from 'react'

import Form from './components/Form'

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Form/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

adding form in JSX

//this is Form.js

import React,{ Component} from 'react'

import './myStyles.css';

class Form extends Component{

    render(){

        return(

            <form>

                <div className = "primary">

                    <label>Username</label>

                    <input type = "text"/>

                </div>

            </form>

        )

    }

}

export default Form

//this is myStyles.css

.primary {

    color:olive;

}

//this is App.js

import React,{Component} from 'react'

import Form from './components/Form'

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Form/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

from is a regular HTML.it is not a controlled React Component.to convert this in to controlled component we need to follow two steps...

The 1st step is to create a component state that will controll the value of the in put element.

Handling onChange events

//this is Form.js

import React,{ Component} from 'react'

import './myStyles.css';

class Form extends Component{

    constructor(props){

        super(props)

        this.state = {

            username:""

        }

    }

    handleUsernameChange = (event)=>{

     this.setState({

         username:event.target.value

     })

    }

    render(){

        return(

            <form>

                <div className = "primary">

                    <label>Username</label>

                    <input type = "text" value = {this.state.username} onChange = {this.handleUsernameChange}/>

                </div>

            </form>

        )

    }

}

export default Form

//this is myStyles.css

.primary {

    color:olive;

}

//this is App.js

import React,{Component} from 'react'

import Form from './components/Form'

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Form/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

controlled components for textArea aswell as select tag.

this will also help you get used to the controll component range  working with form elements..3 simple steps.

1.add the elememt HTML.

2.assign the component state  to the element value.

3.and assign onchange handler that updates the states.

An input element,textArea & select tag are controlled components in React.

submit form data

//this is Form.js

import React,{ Component} from 'react'

import './myStyles.css';

class Form extends Component{

    constructor(props){

        super(props)

        this.state = {

            username:"",

            comments:"",

            topic:"react",

        }

    }

    handleUsernameChange = (event)=>{

     this.setState({

         username:event.target.value

     })

    }

    handleCommentsChange = (event)=>{

        this.setState({

            comments:event.target.value

        })

    }

    handleTopicChange = (event)=>{

       this.setState({

          topic:event.target.value

       })

    }

    handleSubmit = (event)=>{

    alert(`${this.state.username} ${this.state.comments} ${this.state.topic}`)

    //event.preventDefault()

    }

    render(){

        return(

            <form onSubmit = {this.handleSubmit}>

                <div className = "primary">

                    <label>Username</label>

                    <input type = "text" value = {this.state.username} onChange = {this.handleUsernameChange}/>

                </div>

                <div className = "primary">

                    <label>Comments</label>

                    <textarea value ={this.state.comments} onChange = {this.handleCommentsChange}></textarea>

                </div>

                <div className = "primary">

                    <label>Topic:</label>

                    <select value = {this.state.topic} onChange = {this.handleTopicChange}>

                        <option value = "react" className = "primary">React</option>

                        <option value = "angular">Angular</option>

                        <option value = "vue">Vue</option>

                    </select>

                </div>

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

            </form>

        )

    }

}

export default Form

//this is myStyles.css

.primary {

    color:tomato;

}

//this is App.js

import React,{Component} from 'react'

import Form from './components/Form'

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Form/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Destructuring Form

//this is Form.js

import React,{ Component} from 'react'

import './myStyles.css';

class Form extends Component{

    constructor(props){

        super(props)

        this.state = {

            username:"",

            comments:"",

            topic:"react",

        }

    }

    handleUsernameChange = (event)=>{

     this.setState({

         username:event.target.value

     })

    }

    handleCommentsChange = (event)=>{

        this.setState({

            comments:event.target.value

        })

    }

    handleTopicChange = (event)=>{

       this.setState({

          topic:event.target.value

       })

    }

    handleSubmit = (event)=>{

    alert(`${this.state.username} ${this.state.comments} ${this.state.topic}`)

    //event.preventDefault()

    }

    render(){

        const {username,comments,topic}  = this.state

        return(

            <form onSubmit = {this.handleSubmit}>

                <div className = "primary">

                    <label>Username</label>

                    <input type = "text" value = {username} onChange = {this.handleUsernameChange}/>

                </div>

                <div className = "primary">

                    <label>Comments</label>

                    <textarea value ={comments} onChange = {this.handleCommentsChange}></textarea>

                </div>

                <div className = "primary">

                    <label>Topic:</label>

                    <select value = {topic} onChange = {this.handleTopicChange}>

                        <option value = "react" className = "primary">React</option>

                        <option value = "angular">Angular</option>

                        <option value = "vue">Vue</option>

                    </select>

                </div>

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

            </form>

        )

    }

}

export default Form

//this is myStyles.css

.primary {

    color:tomato;

}

//this is App.js

import React,{Component} from 'react'

import Form from './components/Form'

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <Form/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

life cycle methods.

these cannot exist with functional components..they only confined with the class components..

Mounting

1)Mounting---when an instance of a component is being created and inserted into the DOM.

Mounting 4 methods

constuctor, static getDerivedStateFromProps, render and componentDidMount.

Mounting

constuctor(props) it is a special function that will get called whenever a new component is created.

constructor is used for :constructor is perfect for initialising state or binding the event handler to the class instance.

what you should'nt do in a constructor is cause side effects :you should make HTTP requests with in constructor.

when you are defining your own constructor :is that you have to call special function called super().this will call the base class constructor.in our component we have access to this.props only after we are initially called super as props as the argument.

2nd the constructor is also the only place where your expected to change setState() by directly overWriting this.state fields in our other scenarios you have to use this.setState().

constructor :set initial state, find event handler & don't cause any side effects like making ajax calls for example.

static getDerivedStateFromProps(props,state)

React documentation classifies this method as rarely used lifeCycle method. this method is basically used when the state of the component depends on changes in props over time.

Let’s say you have the component, initial state of the component depends on the props being passed to the component in such a scenario we can use this method to set the state.

since this method is static method it does not have access to this keyWord.so you cannot call this.setState() with in this particular method. instead you simply have to return an object that represents new state of the component

what you should'nt do cause side effects: fetching data from the end point so getDerivedStateFromProps use it when state depends on changes in props over time and also do not cause any side effects.

render()

render method is the only require method in a class component.in render method we simply read this.props & this.state & return the JSX which describes the using

render function is a function for the given props and states it should always render the same UI.

we should not: changing the state of component or interacting with the DOM for making any ajax calls.

since it is the render method JSX which also contains the other children component write after the parent render method the children components life cycle methods are also executed.

render() read props and states and return the JSX...

componentDidMount

this method will call only once in the whole lifeCycle of given component and it is invoked immediately after the component to and all children component have been render to the DOM

this method is the perfect place to cause side effects you can interact with DOM for perform any AJAX calls to load Data.

so componentDidMount is perfect place to perform initialisation that requires DOM nodes and also load data by making network requests.

//this is LifecycleA.js

import React,{Component} from 'react'

class LifecycleA extends Component{

    constructor(){

        super()

        this.state = {

           name: 'karthik yadav'

        }

        console.log('LifecycleA constructor')

    }

    static getDerivedStateFromPrpos(props,state){

        console.log('LifecycleA getDerivedStateFromProps')

        return null

    }

    componentDidMount(){

        console.log('LifecycleA componentDidMount')

    }

    render(){

        console.log('LifecycleA render')

        return <div>LifecycleA method</div>

    }

}

export default LifecycleA

//this is App.js

import React,{Component} from 'react';

import LifecycleA from './components/LifecycleA';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <LifecycleA/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is LifecycleA.js

import React,{Component} from 'react'

import LifecycleB from './LifecycleB'

class LifecycleA extends Component{

    constructor(){

        super()

        this.state = {

           name: 'karthik yadav'

        }

        console.log('LifecycleA constructor')

    }

    static getDerivedStateFromPrpos(props,state){

        console.log('LifecycleA getDerivedStateFromProps')

        return null

    }

    componentDidMount(){

        console.log('LifecycleA componentDidMount')

    }

    render(){

        console.log('LifecycleA render')

        return(

        <div>

    <div>LifecycleA method</div>

    <LifecycleB/>

        </div>)

    }

}

export default LifecycleA

//this is LifecycleB.js

import React,{Component} from 'react'

class LifecycleB extends Component{

    constructor(){

        super()

        this.state = {

           name: 'karthik yadav'

        }

        console.log('LifecycleB constructor')

    }

    static getDerivedStateFromPrpos(props,state){

        console.log('LifecycleB getDerivedStateFromProps')

        return null

    }

    componentDidMount(){

        console.log('LifecycleB componentDidMount')

    }

    render(){

        console.log('LifecycleA render')

        return <div>Lifecycle B method</div>

    }

}

export default LifecycleB

//this is App.js

import React,{Component} from 'react';

import LifecycleA from './components/LifecycleA';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <LifecycleA/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Updating

Updating

when the component being Re-rendered that is because of changes to either props or states

5 methods, 3 are falling into the category of rarely used method

static getDerivedStateFromProps(props, state):it takes props and states as it's parameters and has to return either null or an object that represents

updated state of the component.this is method called everytime a component is Re-rendered.this method is used when the states depends on the props of the component

do not cause any side effects:example HTTP request.

rarely udsed method in updating phase

shouldComponentUpdate(nextProps, nextState):this method recieves updated props and states and the purpose of this method it dictates if the component

should re-render or not.by default all the class componets render when the props and states changes.this method can prevent that default behaviour by returning false.

comapre the existing props and state values with the nextProps and state values and return true or false to let react know wether the component should update or not.

this method is for performance optimization.

do not cause side effects. ex:HTTP requests and calling the setState()

render():read the this.props and this.states and return the JSX and disturb the UI.avoid changing the states and interact with DOM in the render() making AJAX calls

getSnapshotBeforeUpdate(prevProps, prevStates):it is called right before the changes from virtual DOM are to be reflected in the DOM.it is rarely used method.you would use

this method to capture some infromation from DOM.ex:u can read the user scroll position and the after the update maintain the scroll position by perfroming some caleculations.

this method either return a null or return a value.return value will be passed as the third parameter to the next method.

getSnapshotBeforeUpdate is used read the current DOM state and return a value for now.

componentDidUpdate(prevProps, prevStates, snapshot):this method is called after the render is finished in the Re-render cycle.component properly render itself after the update

this method call only once in Re-render cycle.

side effects:you can make AJAX calls by comparing the new props and prevProps and then decide wether make AJAX calls or not.

//this is LifecycleA.js

import React,{Component} from 'react'

import LifecycleB from './LifecycleB'

class LifecycleA extends Component{

    constructor(){

        super()

        this.state = {

           name: 'karthik yadav'

        }

        console.log('LifecycleA constructor')

    }

    static getDerivedStateFromPrpos(props,state){

        console.log('LifecycleA getDerivedStateFromProps')

        return null

    }

     shouldComponentUpdate(){

        console.log('LifecycleA shouldComponentUpdate')

        return true

    }

    getSnapshotBeforeUpdate(prevProps, prevState){

        console.log('LifecycleA getSnapshotBeforeUpdate')

        return null

    }

    componentDidUpdate(){

        console.log('LifecycleA componentDidUpdate')

    }

    changeState = () =>{

        this.setState({

            name:'lifEvolution'

        })

    }

    render(){

        console.log('LifecycleA render')

        return(

        <div>

    <div>LifecycleA method</div>

    <button onClick = {this.changeState}>change state</button>

    <LifecycleB/>

        </div>)

    }

}

export default LifecycleA

//this is LifecycleB.js

import React,{Component} from 'react'

class LifecycleB extends Component{

    constructor(){

        super()

        this.state = {

           name: 'karthik yadav'

        }

        console.log('LifecycleB constructor')

    }

    static getDerivedStateFromPrpos(props,state){

        console.log('LifecycleB getDerivedStateFromProps')

        return null

    }

    shouldComponentUpdate(){

        console.log('LifecycleB shouldComponentUpdate')

        return true

    }

    getSnapshotBeforeUpdate(prevProps, prevState){

        console.log('LifecycleB getSnapshotBeforeUpdate')

        return null

    }

    componentDidUpdate(){

        console.log('LifecycleB componentDidUpdate')

    }

    render(){

        console.log('LifecycleB render')

        return <div>Lifecycle B method</div>

    }

}

export default LifecycleB

//this is App.js

import React,{Component} from 'react';

import LifecycleA from './components/LifecycleA';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <LifecycleA/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fragments

Fragments basically let you group a list of children elements without adding extra nodes to the DOM

//this is FragmentDemo.js

import React from 'react';

function FragmentDemo(){

    /\*return(

      <>

        <h1>

          Fragment Demo

        </h1>

        <p> this describes the fragment demo component</p>

        </>

    )\*/

    return(

      <React.Fragment>

        <h1>

          Fragment Demo

        </h1>

        <p>this describes the Fragment Demo component</p>

      </React.Fragment>

    )

}

export default FragmentDemo

//this is App.js

import React,{Component} from 'react';

import FragmentDemo from './components/FragmentDemo';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <FragmentDemo/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Table.js

import React from 'react'

import Columns from './Columns'

function Table(){

    return(

        <table>

            <tbody>

                <tr>

                    <Columns/>

                </tr>

            </tbody>

        </table>

    )

}

export default Table

//this is Columns.js

it is wrong to have <td> element as children in <div>

import React from 'react'

function Columns(){

    return(

        <div>

         <td>Name</td>

         <td>karthik yadav</td>

        </div>

    )

}

export default Columns

//this is App.js

import React,{Component} from 'react';

import Table from './components/Table';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <Table/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Table.js

import React from 'react'

import Columns from './Columns'

function Table(){

    return(

        <table>

            <tbody>

                <tr>

                    <Columns/>

                </tr>

            </tbody>

        </table>

    )

}

export default Table

//this is Columns.js

<div> tag is replaced by the <React.Fragment>

import React from 'react'

function Columns(){  //it is wrong to have <td> element as children in <div>

    return(

        <React.Fragment>

         <td>Name</td>

         <td>karthik yadav</td>

         </React.Fragment>

    )

}

export default Columns

//this is App.js

import React,{Component} from 'react';

import Table from './components/Table';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <Table/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is Table.js

import React from 'react'

import Columns from './Columns'

function Table(){

    return(

        <table>

            <tbody>

                <tr>

                    <Columns/>

                </tr>

            </tbody>

        </table>

    )

}

export default Table

//this is Columns.js

key attribute is the only attribute that can pass React.Fragment

import React from 'react'

function Columns(){

    const items = []

    return(

        <React.Fragment>

            {

                items.map((items)=>{

                    <React.Fragment key = {items.id}>

                     <h1>Title</h1>

                     <p>{items.title}</p>

                    </React.Fragment>

                })

            }

         <td>Name</td>

         <td>karthik yadav</td>

         </React.Fragment>

    )

}

export default Columns

//this is App.js

import React,{Component} from 'react';

import Table from './components/Table';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

        <Table/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PureComponents in React

//ParentComp.js

import React,{Component} from 'react'

import RegularComp from './RegulerComp'

import PureComp from './PureComp'

class ParentComp extends Component{

    constructor(props){

        super(props)

        this.state = {

        name:'karthik yadav',

        }

    }

    componentDidMount(){

        setInterval(()=>{

            this.setState({

                name:'karthik yadav',

            })

        },2000)

    }

    render(){

console.log(‘\*\*\*\*\*\*\*\*\*Parent Component\*\*\*\*\*\*\*\*\*\*\*\*’)

        return(

            <div>

                ParentComponent

                <RegularComp name = {this.state.name}/>

                <PureComp name = {this.state.name}/>

            </div>

        )

    }

}

export default ParentComp

//this is PureComp.js

import React,{PureComponent} from 'react'

class PureComp extends PureComponent{

    render(){

console.log(‘@@@@@@@Pure Component@@@@@@@@’)

        return(

            <div>

              PureComponent {this.props.name}

            </div>

        )

    }

}

export default PureComp

//this is RegularComp.js

import React,{Component} from 'react';

class RegularCompo extends Component{

    render(){

console.log(‘#############Regular Component###########’)

        return(

            <div>

               Regular Component {this.props.name}

            </div>

        )

    }

}

export default RegularCompo

//this is App.js

import React,{Component} from 'react'

import ParentComp from './components/ParentComp';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

      <ParentComp/>

      </div>

    )

  }

}

export default App

for every two seconds the parent and regular components Re-rendered but not the pure components..

RegularComponent PureComponent

1.RegularComponent doesnot implement A PureComponent implement

the shouldComponentUdate method.it shouldComponentUpdate with shallow

always returns true by default. Props and state comparision

PureComponent

ApureComponent implements shouldComponentUpdate with shallow props and state comparison.

It does a shallow comparison of previousState with currentState & previouProps with currentProps.only shallow comparsion indicate difference the component will Re-render.

PureComponents Class Based Components..

Why should we use PureComponent??

PureComponent by preventing unnecessary renders and give you performance boost in certain scenarios

Ex: we are Rendering list of 50 items why not rendering them it is not required you are going to have a good performance boost.

Key point you should not mutate object or array’s in props or state

Ex: add new item to list don’t mutate it by pushing the item in to list reference to array never changes and because pure components only check for that the component will not Re-render even if there is difference. Always return a new Object or array when dealing with Pure Components.

Shallow comparison with respect to primitive and complex types..

Primitive types for two primitive types A and B like Numbers, strings and Boolean. A shallow comparison B returns true if A and B have the same value or same type.

Ex: A and B are strings with value of ‘karthik yadav’ then it returns true.

Complex types for Objects and Arrays. A shallow comparison B returns true if A and B reference same Object..

Array

Var a = [1,2,3];

Var b = [1,2,3];

Var c = a

Var ab\_eq (a===b); //false Because the both a and b array’s are two

Var ac\_eq (a===c); //true different Objects, even though values of both

arrays are same

Object

Var a = {x:1, y:2};

Var b = {x:1, y:2};

Var c = a;

Var ab\_eq = (a===b); //false

Var ac\_eq = (a===c); //true

parentcomponent will never Re-render if the componentShouldUpdate implemeted..componentShouldUpdate implemeted in PureComponent..

import React,{Component,PureComponent} from 'react'

import RegularComp from './RegulerComp'

import PureComp from './PureComp'

class ParentComp extends PureComponent{

    constructor(props){

        super(props)

        this.state = {

        name:'karthik yadav',

        }

    }

    componentDidMount(){

        setInterval(()=>{

            this.setState({

                name:'karthik yadav',

            })

        },2000)

    }

    render(){

        console.log('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ParentComponent\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

        return(

            <div>

                ParentComponent

                <RegularComp name = {this.state.name}/>

                <PureComp name = {this.state.name}/>

            </div>

        )

    }

}

export default ParentComp

//this is PureComp.js

import React,{PureComponent} from 'react'

class PureComp extends PureComponent{

    render(){

console.log(‘@@@@@@@Pure Component@@@@@@@@’)

        return(

            <div>

              PureComponent {this.props.name}

            </div>

        )

    }

}

export default PureComp

//this is RegularComp.js

import React,{Component} from 'react';

class RegularCompo extends Component{

    render(){

console.log(‘#############Regular Component###########’)

        return(

            <div>

               Regular Component {this.props.name}

            </div>

        )

    }

}

export default RegularCompo

//this is App.js

import React,{Component} from 'react'

import ParentComp from './components/ParentComp';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = 'App'>

      <ParentComp/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

React.memo

React.memo is functional components based

//parentComp.js

import React,{Component} from 'react';

import MemoCompo from './MemoCompo';

class ParentComp extends Component{

    constructor(props){

        super(props)

        this.state = {

            name:"karthik yadav",

        }

    }

    componentDidMount(){ ParentComponent is Re-rending after every 10

        setInterval(()=>{ sec.but MemoComponent is not rendering because

            this.setState({ props and states not

                name:"karthik yadav", changed..shouldComponentUpdate will

            }) implement if props and states changes

        },10000)

        }

    render(){

        console.log("\*\*\*\*\*\*\*\*\*\*\*ParentComponent\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

        return(

            <div>

                ParentComponent

                <MemoCompo name = {this.state.name}/>

            </div>

        )

    }

}

export default ParentComp

//this is MemoCompo.js

import React from 'react'

function MemoCompo({name}){

    console.log('Rendering MemoComponent');

    return(

        <div>

            {name}

        </div>

    )

}

export default React.memo(MemoCompo)

//this is App.js

import React from 'react'

import ParentComp from './components/ParentComp';

import './App.css';

function App(){

  return(

    <div className = 'App'>

      <ParentComp/>

    </div>

  )

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Refs

Basically refs make it possible to Access DOM node directly within React

//this is RefsDemo.js

import React,{Component} from 'react'

class RefsDemo extends Component{

    constructor(props){

        super(props)

        this.inputRef = React.createRef()

    }

    componentDidMount(){

        this.inputRef.current.focus()

        console.log(this.inputRef);

    }

    clickHandler = ()=>{

        alert(this.inputRef.current.value)

    }

    render(){

        return(

            <div>

            <input type = "text" ref = {this.inputRef}/>

            <button onClick = {this.clickHandler}>click</button>

            </div>

        )

    }

}

export default RefsDemo

//this is App.js

import React,{Component} from 'react';

import RefsDemo from './components/RefsDemo';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <RefsDemo/>

      </div>

    )

  }

}

export default App

//this is RefsDemo.js

import React,{Component} from 'react'

class RefsDemo extends Component{

    constructor(props){

        super(props)

        this.inputRef = React.createRef()

        this.cbRef = null

        this.setcbRef = (element)=>{

            this.cbRef = element

        }

    }

    componentDidMount(){

        if(this.cbRef){

            this.cbRef.focus()

        }

    }

    clickHandler = ()=>{

        alert(this.inputRef.current.value)

    }

    render(){

        return(

            <div>

            <input type = "text" ref = {this.inputRef}/>

            <input type = "text" ref = {this.setcbRef}/>

            <button onClick = {this.clickHandler}>click</button>

            </div>

        )

    }

}

export default RefsDemo

//this is App.js

import React,{Component} from 'react';

import RefsDemo from './components/RefsDemo';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <RefsDemo/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Refs with class component

Refs are always with classComponent..

1.Creating ref using createRef()

2.Attaching Ref to Input Component.

3.Add a click Handler to the button.and with in clickHandler call child component method using the Ref..

//this is FocusInput.js

import React,{Component} from 'react'

import Input from './Input'

class FocusInput extends Component{

    constructor(props){

        super(props)

        this.componentRef = React.createRef()

    }

    clickHandler  = ()=>{

        this.componentRef.current.focusInput()

    }

    render(){

        return(

            <div>

            <Input ref ={this.componentRef} />

            <button onClick = {this.clickHandler}>Focus Input</button>

            </div>

        )

    }

}

export default FocusInput

//this is Input.js

import React,{Component} from 'react'

class Input extends Component{

    constructor(props){

        super(props)

        this.inputRef = React.createRef()

    }

    focusInput(){

        this.inputRef.current.focus()

    }

    render(){

        return(

            <div>

            <input type = "text" ref = {this.inputRef}/>

            </div>

        )

    }

}

export default Input

//this is App.js

import React,{Component} from 'react';

import FocusInput from './components/FocusInput';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <FocusInput/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Forwarding Refs

Automatically passing Refs through component to one of its children.

//this is FRParentInput.js

import React,{Component} from 'react'

import FRInput from './FRInput'

class FRParentInput extends Component{

    constructor(props){

        super(props)

        this.inputRef = React.createRef()

    }

    clickHandler = ()=>{

        this.inputRef.current.focus()

    }

    render(){

        return(

            <div>

               <FRInput ref = {this.inputRef}/>

               <button onClick = {this.clickHandler}>Focus Input</button>

            </div>

        )

    }

}

export default FRParentInput

//this is FRInput.js

import React from 'react'

/\*function FRInput(){

    return(

        <div>

          <input type = "text"/>

        </div>

    )

}\*/

const FRInput =React.forwardRef((props,ref)=>{

    return(

        <div>

            <input type = "text" ref = {ref}/>

        </div>

    )

})

export default FRInput

//this is App.js

import React,{Component} from 'react';

import FRParentInput from './components/FRParentInput';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <FRParentInput/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

React portals

React portals way to provide children’s into a DOM node that exists outside the DOM hydroid key of the parent component.

The ability to break out this DOM tree.

You can render component on this DOM node that is not under this root element.

1.One DOM element in HTML that we were mounting React application into<div className = "App">

2.in index.HTML we can see the element with element <div id="root"></div>

3.index.js we use ReactDOM.render ()ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);mounting our App component on to a root element.

In browser every single React Component in our application falls under the root element.i.e<div id = ‘root’>.

React portal provides the ability to break out the DOM tree.so you can render a component on DOM node that is not under root element.

//this is PortalDemo.js

import React from 'react';

function PortalDemo(){

    return(

        <div>

       Portals Demo

        </div>

    )

}

export default PortalDemo

//this is index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <!--

      manifest.json provides metadata used when your web app is installed on a

      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

    -->

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <!--

      Notice the use of %PUBLIC\_URL% in the tags above.

      It will be replaced with the URL of the `public` folder during the build.

      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

      work correctly both with client-side routing and a non-root public URL.

      Learn how to configure a non-root public URL by running `npm run build`.

    -->

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

    <div id = "portal-root"></div>note: the element is falls under the root element.not under the portal root element.

    <!--

      This HTML file is a template.

      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.

      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.

      To create a production bundle, use `npm run build` or `yarn build`.

    -->

  </body>

</html>

//this is App.js

import React,{Component} from 'react';

import PortalDemo from './components/PortalDemo';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <PortalDemo/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//this is PortalDemo.js

<h1> tag is under the portal root DOM node and not the root DOM node.on reac

t application all the components are children to app component and the app

component is mounted on to the root DOM node it is possible to break away from that and mount on any DOM node that you wish to using React portals

1st parameter to create portals can be any element that React can render numbers, strings, JSX, even Components.

import React from 'react';

import ReactDOM from 'react-dom';

function PortalDemo(){

    return ReactDOM.createPortal(

        <h1>

       Portals Demo

        </h1>,

        document.getElementById('portal-root')

    )

}

export default PortalDemo

//this is index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <!--

      manifest.json provides metadata used when your web app is installed on a

      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

    -->

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <!--

      Notice the use of %PUBLIC\_URL% in the tags above.

      It will be replaced with the URL of the `public` folder during the build.

      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

      work correctly both with client-side routing and a non-root public URL.

      Learn how to configure a non-root public URL by running `npm run build`.

    -->

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

    <div id = "portal-root"></div>

    <!--

      This HTML file is a template.

      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.

      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.

      To create a production bundle, use `npm run build` or `yarn build`.

    -->

  </body>

</html>

//this is App.js

import React,{Component} from 'react';

import PortalDemo from './components/PortalDemo';

import './App.css';

class App extends Component{

  render(){

    return(

      <div className = "App">

        <PortalDemo/>

      </div>

    )

  }

}

export default App

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Why do we need portals

1.having to deal with parent component css when the child component is modal a pop-up or a tool tip.