**Form Validation**

App.js

import './App.css';

import Login from './Login';

import React from 'react';

function App(){

    return(

        <div className="App">

            < Login />

        </div>

    );

}

export default App;

Login.js:- export App.js

import { useState } from "react";

function Login() {

    const [name,setName]=useState("")

    const [password,setPassword]=useState("")

    const [nerror,setNerror]=useState(false)

    const [perror,setPerror]=useState(false)

    function Submit(form) {

        if(name.length<=3 || password.length<=3){

            alert("Please Enter Your Valid registration")

        }

        else{

            alert("Good Job :)")

        }

        form.preventDefault()

    }

    function Name(form){

        let data=form.target.value

        if(data.length<3){

            setNerror(true)

        }

        else{

            setNerror(false)

        }

        setName(data)

    }

    function Password(form){

        let data=form.target.value

        if(data.length<3){

            setPerror(true)

        }

        else{

            setPerror(false)

        }

        setPassword(data)

    }

    return(

        <div>

            <h1>My Form</h1>

            <form onSubmit={Submit}>

                <input type="text" placeholder="Enter Name" onChange={Name}/>

                {nerror?<p>Invalid</p>:""}

                <br/><br/>

                <input type="text" placeholder="Enter Password" onChange={Password}/>

                {perror?<p>Invalid</p>:""}

                <br/><br/>

                <button>Submit</button>

            </form>

        </div>

    );

}

export default Login;

Pass function in props

App.js:-

import './App.css';

import User from './User';

import Login from './Login';

import React from 'react';

function App(){

    function Police() {

        alert("AAYA POLICE!")

    }

    return(

        <div className="App">

            <User  data={Police}/>

            <Login data={Police}/>

        </div>

    );

}

export default App;

User.js:- export on App.js like, User component.

import React from "react";

function User(props) {

    return(

        <div>

            <h1>hello bhai hello!</h1>

            <button onClick={()=>props.data()}>Call THe Police</button>

        </div>

    );

}

export default User;

Login.js:- there’s second component of Login.js and they export App.js.

import React from "react";

function Login(props){

    return(

        <div style={{marginTop:50}}>

            <button onClick={()=>props.data()}>hello</button>

        </div>

    );

}

export default Login;

Mapping Data

import React, { Component } from 'react';

class App extends React.Component {

 constructor() {

      super();

      this.state = {

         data:

         [

            {

               "name":"Abhishek"

            },

            {

               "name":"Saharsh"

            },

            {

               "name":"Ajay"

            }

         ]

      }

   }

   render() {

      return (

         <div>

            <StudentName/>

            <ul>

                {this.state.data.map((item) => <List data = {item} />)}

            </ul>

         </div>

      );

   }

}

class StudentName extends React.Component {

   render() {

      return (

         <div>

            <h1>Student Name Detail</h1>

         </div>

      );

   }

}

class List extends React.Component {

   render() {

      return (

         <ul>

            <li>{this.props.data.name}</li>

         </ul>

      );

   }

}

export default App;

Class Component How To Use Of State And Got Data

import React, { Component } from 'react';

class App extends React.Component {

 constructor() {

      super();

      this.state = { displayBio: false };

      }

      render() {

          const bio = this.state.displayBio ? (

              <div>

                  <p><h3>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad. We have a team of experienced Java developers and trainers from multinational companies to teach our campus students.</h3></p>

            </div>

              ) : null;

              return (

                  <div>

                      <h1> Welcome to JavaTpoint!! </h1>

                      { bio }

                  </div>

              );

     }

}

export default App;

Change The State Data

import React, { Component } from 'react';

class App extends React.Component {

 constructor() {

      super();

      this.state = { displayBio: false };

      console.log('Component this', this);

      this.toggleDisplayBio = this.toggleDisplayBio.bind(this);

      }

      toggleDisplayBio(){

          this.setState({displayBio: !this.state.displayBio});

          }

      render() {

          return (

              <div>

                  <h1>Welcome to JavaTpoint!!</h1>

                  {

                      this.state.displayBio ? (

                          <div>

                              <p><h4>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad. We have a team of experienced Java developers and trainers from multinational companies to teach our campus students.</h4></p>

                              <button onClick={this.toggleDisplayBio}> Show Less </button>

                        </div>

                          ) : (

                              <div>

                                  <button onClick={this.toggleDisplayBio}> Read More </button>

                              </div>

                          )

                  }

             </div>

        )

    }

}

export default App;

State and Props

import React, { Component } from 'react';

class App extends React.Component {

   constructor(props) {

      super(props);

      this.state = {

         name: "JavaTpoint",

      }

   }

   render() {

      return (

         <div>

            <JTP jtpProp = {this.state.name}/>

         </div>

      );

   }

}

class JTP extends React.Component {

   render() {

      return (

          <div>

              <h1>State & Props Example</h1>

              <h3>Welcome to {this.props.jtpProp}</h3>

              <p>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad.</p>

         </div>

      );

   }

}

export default App;

React Props Validation

|  |  |  |
| --- | --- | --- |
| **SN** | **PropsType** | **Description** |
| **1.** | PropTypes.any | The props can be of any data type. |
| **2.** | PropTypes.array | The props should be an array. |
| **3.** | PropTypes.bool | The props should be a boolean. |
| **4.** | PropTypes.func | The props should be a function. |
| **5.** | PropTypes.number | The props should be a number. |
| **6.** | PropTypes.object | The props should be an object. |
| **7.** | PropTypes.string | The props should be a string. |
| **8.** | PropTypes.symbol | The props should be a symbol. |
| **9.** | PropTypes.instanceOf | The props should be an instance of a particular JavaScript class. |
| **10.** | PropTypes.isRequired | The props must be provided. |
| **11.** | PropTypes.element | The props must be an element. |
| **12.** | PropTypes.node | The props can render anything: numbers, strings, elements or an array (or fragment) containing these types. |
| **13.** | PropTypes.oneOf() | The props should be one of several types of specific values. |
| **14.** | PropTypes.oneOfType([PropTypes.string,PropTypes.number]) | The props should be an object that could be one of many types. |

import React, { Component } from 'react';

import PropTypes from 'prop-types';

class App extends React.Component {

   render() {

      return (

          <div>

              <h1>ReactJS Props validation example</h1>

              <table>

                  <tr>

                      <th>Type</th>

                      <th>Value</th>

                      <th>Valid</th>

                  </tr>

                <tr>

                      <td>Array</td>

                      <td>{this.props.propArray}</td>

                      <td>{this.props.propArray ? "true" : "False"}</td>

                  </tr>

                  <tr>

                      <td>Boolean</td>

                      <td>{this.props.propBool ? "true" : "False"}</td>

                      <td>{this.props.propBool ? "true" : "False"}</td>

                  </tr>

                  <tr>

                      <td>Function</td>

                      <td>{this.props.propFunc(5)}</td>

                      <td>{this.props.propFunc(5) ? "true" : "False"}</td>

                  </tr>

                  <tr>

                      <td>String</td>

                      <td>{this.props.propString}</td>

                      <td>{this.props.propString ? "true" : "False"}</td>

                  </tr>

                  <tr>

                      <td>Number</td>

                      <td>{this.props.propNumber}</td>

                      <td>{this.props.propNumber ? "true" : "False"}</td>

                  </tr>

             </table>

        </div>

        );

   }

}

App.propTypes = {

    propArray: PropTypes.array.isRequired,

    propBool: PropTypes.bool.isRequired,

    propFunc: PropTypes.func,

    propNumber: PropTypes.number,

    propString: PropTypes.string,

}

App.defaultProps = {

    propArray: [1,2,3,4,5],

    propBool: true,

    propFunc: function(x){return x+5},

    propNumber: 1,

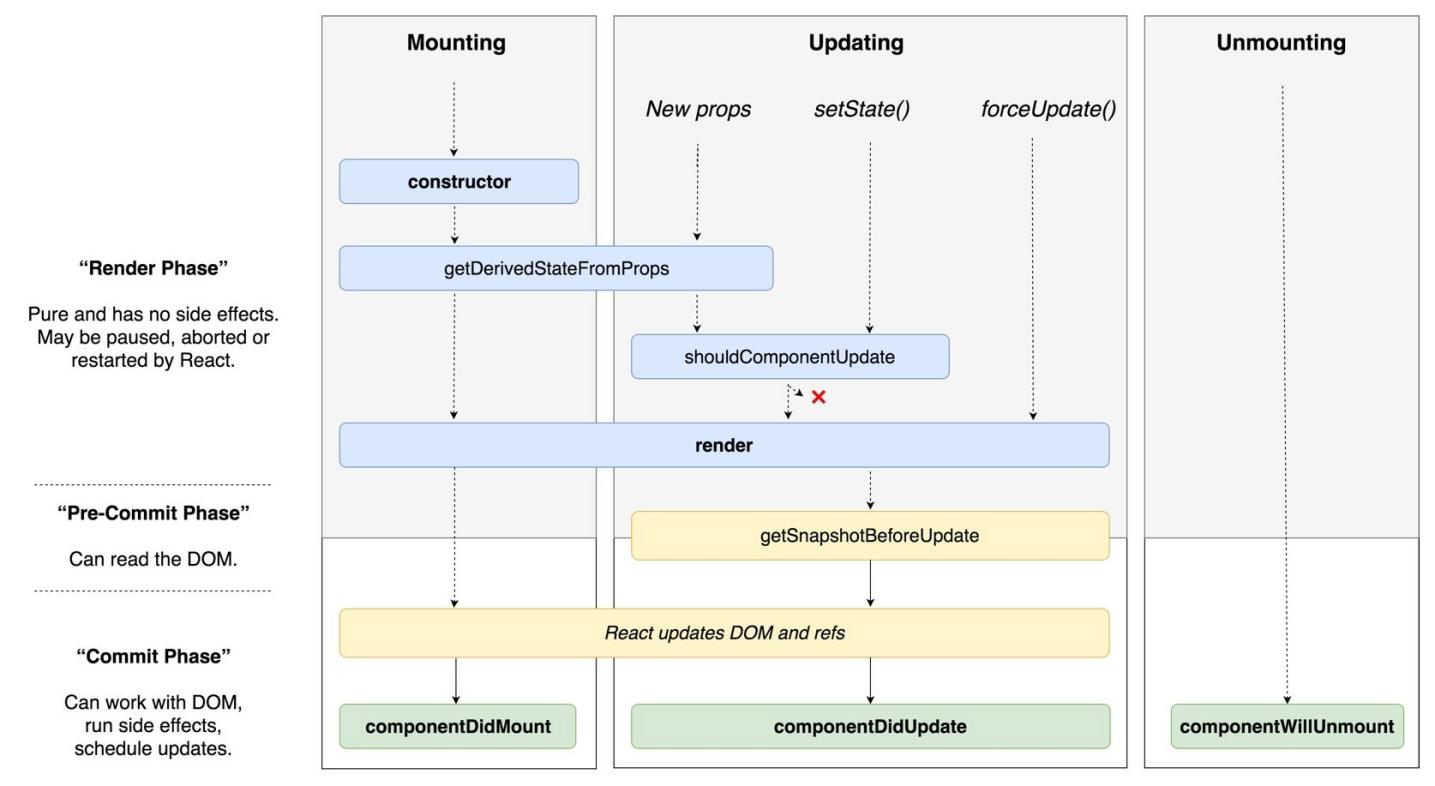
    propString: "JavaTpoint",

}

export default App;

Multi-function of using behind props.

Life Cycle Of React.js



There life cycle methods is multiple but phase is three:-

1. Mounting: - it’s birth of cycle or create/start the cycle.
2. Updating: - it’s time to time update any situations, any conditions are there.
3. Unmounting: - it’s expire of cycle or end the cycle.

**Mounting**

1. **Constructor: - there’s method it’s ready before html tag’s and when we are create the components so started first part to create constructor and call first.**

**App.js**

import './App.css';

import React from 'react';

import react from 'react';

class App extends React.Component{

    constructor(){

        super(); // it's used to parents class are call ( React.Component  class of parent so it's must called)

        console.warn("constructor")

    }

    render(){

        console.warn("render")

        return(

            <div className="App">

                <h1>hello guy's</h1>

            </div>

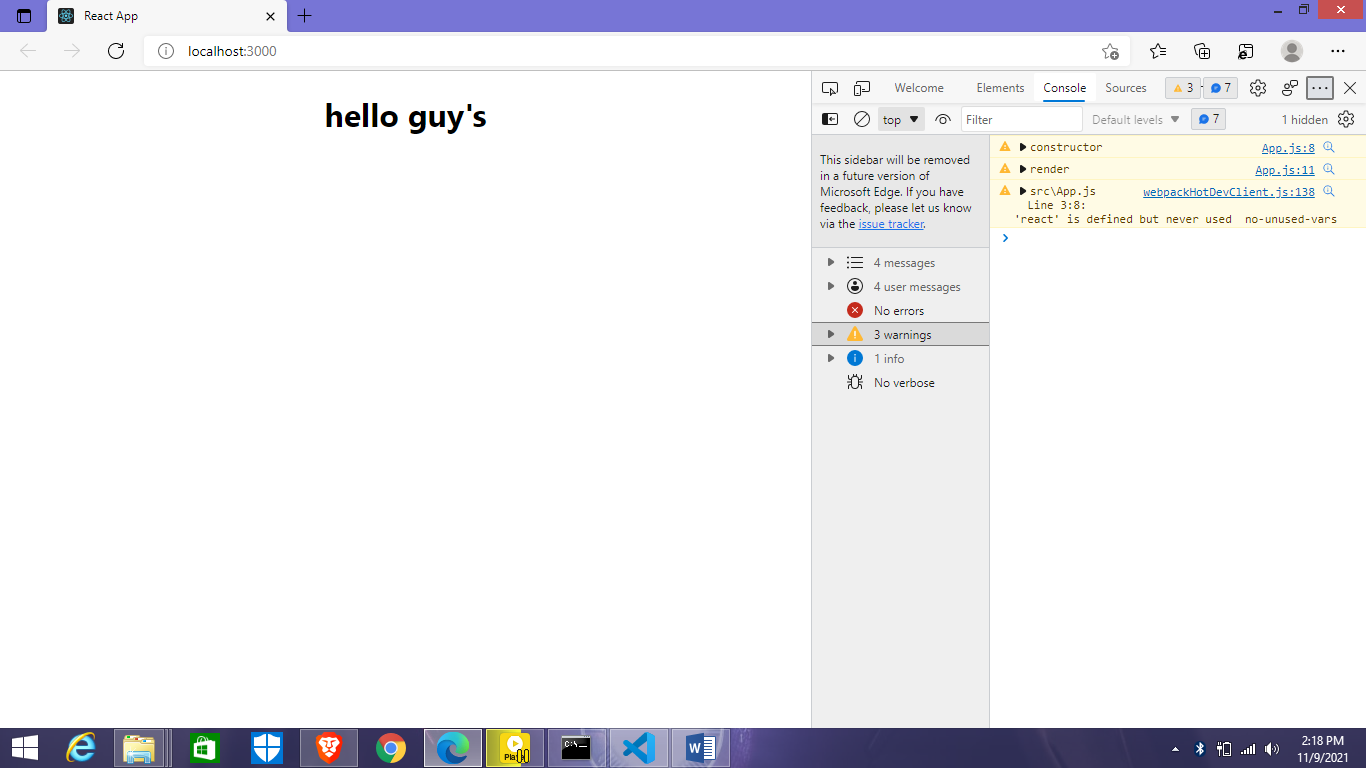
        );

    }

}

export default App;

**Like this:-**

****

import './App.css';

import React from 'react';

import react from 'react';

class App extends React.Component{

    constructor(){

        super(); // it's used to parents class are call ( React.Component  class of parent so it's must called)

        this.state={

            data:"lucky"// now create state like this.

        }

    }

    render(){ // it’s mandatory from create an class.

        return(

            <div className="App">

                <h1>hello guy's {this.state.data}</h1>// And state call like this.

            </div>

        );

    }

}

export default App;

1. **Render: - It’s used for classes, basically render is executed and then represent class functions.**

**Mostly render is call up to components are ready and second state or props will update.**

**App.js:- it’s run with Props**

import './App.css';

import React,{ useState } from 'react';

import User from './User';

function App(){

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

    return(

        <div className="App">

            <h1>Hello!</h1>

            <User name={name}/>

            <button onClick={()=>setName("bandhey")}>Bye!</button>

        </div>

    )

}

export default App;

**User.js:-**

import React from "react";

class User extends React.Component{

    render()

    {

        console.warn("render method ", this.props)

        return(

            <div className="App">

                <h1>Hello bhai! {this.props.name}</h1>

            </div>

        );

    }

}

export default User;