**React JS**

* **Babel**
* Babel is a JavaScript compiler that converts modern JavaScript code into a version compatible with all browsers. Babel enables React developers to use the latest JavaScript syntax in their components. Babel transpiles modern JavaScript for use in React components and all browsers.
* **What is Component**
* Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.
* Components let you split the UI into independent, reusable pieces, and think about each piece in isolation. This page provides an introduction to the idea of components. Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen.
* Components are used to divide our pages UI into small piece and when we want to isolate our code and it does not effect on other component this is called component.
* **Function Component Router in React JS**
* **MainRouter.jsx**

import React, { Suspense } from "react";

import { createBrowserRouter } from "react-router-dom";

// import { Link } from "react-router-dom";

import Main from "../Pages/Main";

import Contact from "../Pages/Contact";

import About from "../Pages/About";

import HeaderFile from "../Component/HeaderFile";

import Example from "../Pages/Example";

const Functionalconporoute = React.lazy(() => { return import('./Functionalconporoute.jsx') })

const Classcomporoute = React.lazy(() => { return import('./Classcomporoute.jsx') })

const MainRouter = createBrowserRouter([

  {

    path: "/",

    element: (

      <>

        <HeaderFile />

        <Main />

        {/\* <Link to="/"></Link> \*/}

      </>

    ),

  },

  {

    path: "/about",

    element: (

      <>

        <HeaderFile />

        <About />

        {/\* <Link to="/"></Link> \*/}

      </>

    ),

  },

  {

    path: "/contact",

    element: (

      <>

        <HeaderFile />

        <Contact />

        {/\* <Link to="/"></Link> \*/}

      </>

    ),

  },

  {

    path: "/example",

    element: (

      <>

        <HeaderFile />

        <Example />

        {/\* <Link to="/"></Link> \*/}

      </>

    ),

    children: [

      {

        path: "/example/functionalcomponent/\*",

        element: <Suspense fallback={<>Loading....</>} ><Functionalconporoute /></Suspense>

      },

    ]

  },

  {

    path: "/example",

    element: (

      <>

        <HeaderFile />

        <Example />

        {/\* <Link to="/"></Link> \*/}

      </>

    ),

    children: [

      {

        path: "/example/classcomponent/\*",

        element: <Suspense fallback={<>Loading....</>} ><Classcomporoute /></Suspense>

      },

    ]

  }

]);

export default MainRouter;

* **Functionalconporoute.jsx**

import React from "react";

import { useRoutes } from "react-router-dom";

import Functioncompomenu from "./../Component/FunctionComponent/01Functioncompomenu.jsx";

import Functioncompointro from "./../Component/FunctionComponent/02Functioncompointro.jsx";

const Functionalconporoute = () => {

    const routes = useRoutes([

        {

            path:"/",

            element:<Functioncompomenu/>,

            children:[

                {

                    path:"functioncompointro",

                    element:<Functioncompointro/>

                }

            ]

        }

    ])

    return routes;

}

export default Functionalconporoute;

* **Functioncompomenu.jsx**

import React from "react";

import { Link, Outlet } from "react-router-dom";

const Functioncompomenu = () => {

    return (<>

        <div className="row">

            <div className="col offset-6">

                <ol>

                    <li><Link to="functioncompointro">Functional Components Intro</Link> </li>

                </ol>

            </div>

        </div>

        <div className="row">

            <div className="col">

                <Outlet></Outlet>

            </div>

        </div>

    </>);

}

export default Functioncompomenu;

* When we want to print child component then we use Outlet.
* **Functioncompointro.jsx**

import React from "react";

const Functioncompointro = () => {

    return ( <>

        <div className="row">

            <div className="col offset-6">

                <h2>Functioncompointro</h2>

            </div>

        </div>

    </> );

}

export default Functioncompointro;

* **What is class component**
* Class component defined with class keyword. It extends the Component. In class component we provide render(),
* We can’t create object here so if we want to access some functionality we use render.
* Render() basically return functionality it always written JSX.
* Here render means when we use class component if we want to return anything we write render function. This is class component so class component doesn’t return function expect function component but when we write render it allows returning in class component. Without render it gives error.
* **Class component Router**
* Main router file is same as function component route.
* After lazy loading of classcomporoute file it is same as function component route.
* **Classcompomenu.jsx**

import React from "react";

import { Link, Outlet } from "react-router-dom";

import { Component } from "react";

class Classcompomenu extends Component {

    // state = {}

    render() {

        return (<>

            <div className="row">

                <div className="col">

                    <ol>

                        <li><Link to="classcompointro">Class Components Intro</Link> </li>

                    </ol>

                </div>

            </div>

            <div className="row">

                <div className="col">

                    <Outlet></Outlet>

                </div>

            </div>

        </>);

    }

}

export default Classcompomenu;

* **Classcompointro.jsx**

import React from "react";

import { Component } from "react";

class Classcompointro extends Component {

    // state = {}

    render() {

        return (<>

            <div className="row">

                <div className="col">

                    <h2>Classcompointro</h2>

                </div>

            </div>

        </>);

    }

}

export default Classcompointro;

* **JSX**
* JSX stands for JavaScript XML
* JSX allows us to write your html within JavaScript in very easy manner.
* JSX is an extension of JavaScript that allows you to write HTML in JavaScript file.
* JSX syntax is not understand by browser and so we need transpiler like babel to convert the code from JSX to JavaScript.
* Example of JSX
* When we want to print something in JSX we use { }
* <input type="text" name="" id="" value={name}></input>
* Username: 
* In above Example we pass value in input tag because of Virtual dom we can't be able to edit or delete input tag value.
* **JSX.jsx**

import React from "react";

import { Component } from "react";

class JSX extends Component {

    render() {

        let name = "dishank"

        let input = '<input type="text" name="" id="" value={name}></input>';

        let inputtag = <input type="text" value={name} name="" id=""></input>;

        return (

            <>

                <div className="row">

                    <div className="col-6">

                        <p>JSX stands for JavaScript XML</p>

                        <p>JSX allows us to write your html within JavaScript in very easy manner.</p>

                        <p>JSX is an extension of JavaScript that allows you to write HTML in JavaScript file.</p>

                        <p>JSX syntax is not understand by browser and so we need transpiler like babel to convert the code from JSX to JavaScript.</p>

                        Example of JSX

                        <p>When we want to print something in JSX we use &#123; &#125; </p>

                        <p>{input}</p>

                        <p>Username: {inputtag}</p>

                        <p>In above Example we pass value in input tag because of Virtual dom we can't be able to edit or delete input tag value.</p>

                        <p>6+6 = {6 + 6}</p>

                    </div>

                </div>

            </>

        );

    }

}

export default JSX;

* **Event Syntax**
* Wrong <button onClick={this.event()}>onClick={this.function\_name()}>.
* If we want to call the function then this.functionname() these is a wrong way to call the function in class component.
* {this.functionname} is only right way to call function.

import { Component } from "react";

class Event extends Component {

    event() {

        alert("called");

    }

    render() {

        return (

            <>

                <div className="row">

                    <div className="col-6">

                    </div>

                </div>

                <p>

                    react-dom.development.js:86 Warning: You provided a `value` prop to a form field without an `onChange` handler. This will render a read-only field. If the field should be mutable use `defaultValue`. Otherwise, set either `onChange` or `readOnly <br /><br />

                    &lt;button onclick="event()"&gt;Click onclick="event()"&lt;/button&gt;

                    {/\* <button onclick="kaiPanEvent()">Click onclick="kaiPanEvent()"</button>   \*/}

                </p>

                <br />

                {/\* <button onclick={kaiPanEvent()}>Click onclick=&#123;kaiPanEvent()&#125;</button> \*/}

                {/\* <button onClick={kaiPanEvent}>onclick=&#123;kaiPanEvent&#125; Wrong</button>   \*/}

                Wrong

                &lt;button onClick=&#123;this.event()&#125;&gt;onClick=&#123;this.function\_name()&#125;&gt;

                <p>If we want to call the function then this.functionname() these is a wrong way to call the function in class component</p>

                <br />

                Correct

                <button onClick={this.event}>onClick=&#123;this.function\_name&#125;</button>

                <p>&#123;this.functionname&#125; is only right way to call function</p>

                <br />

                <br />

            </>

        );

    }

}

export default Event;

* **State in Class component**
* Constuctor : constructor is method that invokes by default when object are created.
* Super: Super() function is to call the constructor of the parent class. It is used when we need to access a few variables in the parent class.
* State: State is a React JS variable. It is mutable it means when we want to change the data into runtime then we use state.
* In normal variable we cannot change the value in react so state comes into picture Using the setState we are able to change the value any time.
* two ways to create a function with arrow function we don't require to bind the function whenever we want to change the state value
* If we create a normal function then we need to bind using below syntax:
* this.changeStateData2 = this.changeStateData2.bind(this);

import { Component } from "react";

class StateinClass extends Component {

    // usernamedm = "Data Member"

    constructor() {

        super(); //Must call super constructor in derived class before accessing 'this' or returning from derived constructor

        // this.usernamedm = "Something from DM"

        // this.state = { usernamedm: "Something from DM" }

        // console.log(this.usernamedm);

        // console.log("Constuctor calling");

        this.state = { data: "Test" }

        this.state = { data2: "Test2" }

        // this.clickHandle = this.clickHandle.bind(this);

        this.changeStateData2 = this.changeStateData2.bind(this);

    }

    // databuilding(){ let variable = "some variable" }

    // datachanging(){}

    // var username = "Something"

    // clickHandle = () => {

    //     // this.setState({ usernamedm: "Dishank" })

    //     // console.log(this.usernamedm);

    //     // console.log("checking");

    //     // username = "change"

    // }

    // changeStateData() {

    //     this.setState({ DataMariMarji: "Testing from changeStateData" })

    // }

    changeStateData = () => {

        this.setState({ data: "Change data without bind" })

        console.log(this.state.data);

    }

    changeStateData2() {

        this.setState({ data2: "Change data with bind" })

        console.log(this.state.data2);

    }

    render() {

        // console.log("From Render",this.usernamedm);

        // function clickHandle() {

        //     console.log(this.usernamedm);

        //     // console.log("checking");

        //     // username = "change"

        // }

        // var username = "Something"

        return (

            <>

                <div className="row">

                    <div className="col-6">

                        <p>Constuctor : constructor is method that invokes by default when object are created</p>

                        <p>Super: Super() function is to call the constructor of the parent class. It is used when we need to access a few variables in the parent class. </p>

                        <p>State: State is a React JS variable. It is mutable it means when we want to change the data into runtime then we use state.</p>

                        <p>In normal variable we cannot change the value in react so state comes into picture Using the setState we are able to change the value any time.</p>

                        <p>two ways to create a function with arrow function we don't require to bind the function whenever we want to change the state value</p>

                        <p>If we create a normal function then we need to bind using below syntax:</p>

                        <p>this.changeStateData2 = this.changeStateData2.bind(this);</p>

                        {/\* <button onClick={this.clickHandle}>Click</button>

                <button onClick={() => { console.log("btn Clicked"); }}>Click</button> \*/}

                        {/\* <button onClick={() => { this.setState({ DataMariMarji: "change" }) }}>State change{this.state.DataMariMarji}</button> \*/}

                        <button onClick={this.changeStateData}>Change data without bind</button>

                        <button onClick={this.changeStateData2}>Change data with bind</button>

                    </div>

                </div>

            </>

        );

    }

}

export default StateinClass;