# REACT

* A javascript library for single page applications, maintained by facebook

🡪 SPAs are loaded only once and all the work is done alone by javascript. Navigation from one page to another is done without reloading the page

* React provides a very good user experience as it is raw. Very easy to use and is free from too many concepts or file structures. We can directly use React, if we already know JS.
* Prerequisites🡪Node JS
* npx create-react-app myfirstapp

🡪 npx is bundled with npm and is used to execute uninstalled packages.

🡪 create-react-app is a utility to generate our first React application

* JSX🡪 a syntax extension of javascript
* React [doesn’t require](https://legacy.reactjs.org/docs/react-without-jsx.html) using JSX, but most people find it helpful as a visual aid when working with UI inside the JavaScript code. It also allows React to show more useful error and warning messages.

# RENDERING COMPONENT INSIDE ROOT IN REACT

1. Embedding expression in JSX

App.jx

//import logo from './logo.svg';

import './App.css';

function App() {

  function formatUser(user){

    return user.firstName + " " + user.lastName ;

  }

  const user ={

    firstName : "Namrata",

    lastName: "Das"

  }

  //const element=<h1>Hello All! Welcome to the world of React</h1>

  return (

        <div>

          Hello {formatUser(user)}! Welcome to the world of React

        </div>

  );

}

export default App;

2.  Conditional JSX

function getGreeting(user){

   if(user){

    return <div>Hello {formatUser(user)}!</div>

   }

   else{

    return <div>Hello Stranger!</div>

   }

 }

  function formatUser(user){

    return user.firstName + " " + user.lastName ;

  }

  const user ={

    firstName : "Namrata",

    lastName: "Das"

  }

  //const element=<h1>Hello All! Welcome to the world of React</h1>

  return (

        <div>

          {getGreeting(user)}

        </div>

  );

# RENDERING REACT ELEMENTS INSIDE ROOT

Index.js

const element= <h1> Hello</h1>

root.render(

<>{element}</>

);

**UPDATING RENDERED ELEMENTS**

Index.js

const rootElement = document.getElementById("root");

const root = createRoot(rootElement);

function tick () {

const element= (

<div>

<h1> Hello World</h1>

<h2>It is {new Date().toLocaleTimeString()}</h2>

</div>

)

root.render(element);

}

setInterval(tick, 1000)

# Creating and nesting components

React apps are made out of components. A component is a piece of the UI (user interface) that has its own logic and appearance. A component can be as small as a button, or as large as an entire page.

React components are JavaScript functions that return markup:

function MyButton() {

return (

<button>I'm a button</button>

);

}

Now that you’ve declared MyButton, you can nest it into another component:

export default function MyApp() {

return (

<div>

<h1>Welcome to my app</h1>

<MyButton />

</div>

);

}

Notice that <MyButton /> starts with a capital letter. That’s how you know it’s a React component. React component names must always start with a capital letter, while HTML tags must be lowercase.

# CREATING COMPONENT

1. Create a component folder.
2. Create a file named “Welcome.js” inside Component folder
3. Create your component🡪

function Welcome(){

return <h1> Hello from the Welcome Component </h1>;

}

export default Welcome;

1. Either you can directly add component in index.js 🡪 root.render(<Welcome/>);

OR

Add it to app.js 🡪

function App(){

return <Welcome/>

}

import Welcome from './Component/Welcome';

# PASSING PROPS TO COMPONENT

Welcome.js

function Welcome(props){

    console.log(props);

    return <h1>Hello, {props.user.firstName} from Welcome Componenet</h1>;

}

export default Welcome;

App.js

const userInfo ={

    firstName : "Namrata",

    lastName: "Das"

  }

    return <><Welcome user = {userInfo}/></>

**CREATE REUSABLE COMPONENTS**

App.js

 const userInfo ={

    firstName : "Namrata",

    lastName: "Das"

  }

    return <><Welcome user = "Namrata"/>

            <Welcome user = "Navneet"/>

            <Welcome user = "Nimrit"/></>

}

Welcome.js

function Welcome(props){

    console.log(props);

    return <h1>Hello, {props.user} from Welcome Component</h1>;

}

export default Welcome;

**Labsheet 11 Eg 11**

App.js

return (

              <div>

              <h1>Spin up a simple component to learn how to use react</h1>

              <h2>My First React JS Program</h2>

              </div>

              );

**Labsheet 11 Eg 12**

App.js

const text = <h1>I am a button. Please click me</h1>

return (

<div>

<h2>Welcome to my application</h2>

<MyButton text={text} />

</div>

);

MyButton.js

function MyButton(props) {

    return <button>{props.text}</button>;

    }

export default MyButton;

**Understanding React Components**

React component accomplish these feature using three concepts −

* **Properties** − Enables the component to receive input.
* **Events** − Enable the component to manage DOM events and end-user interaction.
* **State** − Enable the component to stay stateful. Stateful component updates its UI with respect to its state.

**STATE MANAGEMENT IN REACT**

**Without useState**

Clock.js

function Clock(){

    function showDate(){

        console.log(new Date().toString());

        return <h1>{new Date().toString()}</h1>

    }

setInterval(showDate, 1000)

return <div>{showDate()}</div>

}

export default Clock;

App.js

function App(){

 return <><Clock/></>

}

**With useState**

**Eg 1**

function Clock(){

   const [time,setTime]= useState(new Date().toString())

    function showDate(){

        //console.log(new Date().toString());

        setTime(new Date().toString());

    }

setInterval(showDate, 1000)

return <div>{time}</div>

}

export default Clock;

**Eg 2**

***Reading state***

import { useState } from "react";

import ReactDOM from "react-dom/client";

function FavoriteColor() {

const [color, setColor] = useState("red");

return <h1>My favorite color is {color}!</h1>

}

***Updating state***

import React,  {useState} from 'react'

function Color() {

    const [color, setColor] = useState("red");

    return (

    <>

        <h1>My favorite color is {color}!</h1>

        <button type="button" onClick={() => setColor("blue")}>Blue</button>

    </>)

  }

  export default Color

## What Can State Hold

The useState Hook can be used to keep track of strings, numbers, booleans, arrays, objects, and any combination of these!

We could create multiple state Hooks to track individual values.

**Car.js**

function Car() {

    const [brand, setBrand] = useState("Ford");

    const [model, setModel] = useState("Mustang");

    const [year, setYear] = useState("1964");

    const [color, setColor] = useState("red");

    return (

      <>

        <h1>My {brand}</h1>

        <p>

          It is a {color} {model} from {year}.

        </p>

      </>

    )

  }

Or, we can just use one state and include an object instead!

function Car() {

const [car, setCar] = useState({

brand: "Ford",

model: "Mustang",

year: "1964",

color: "red"

});

return (

<>

<h1>My {car.brand}</h1>

<p>

It is a {car.color} {car.model} from {car.year}.

</p>

</>

)

}

## REACT EVENTS

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

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

**Adding Events**

React events are written in camelCase syntax:

onClick instead of onclick.

React event handlers are written inside curly braces:

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

**Football.js**

function Football() {

    const shoot = () => {

      alert("Great Shot!");

    }

    return (

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

    );

  }

  export default Football

## Passing Arguments

To pass an argument to an event handler, use an arrow function.

function Football() {

    const shoot = (a) => {

      alert(a);

    }

    return (

      <button onClick={()=>shoot("Goal")}><h1>Take the shot</h1></button>

    );

  }

  export default Football

## React Event Object

Event handlers have access to the React event that triggered the function.

In our example the event is the "click" event.

function Football() {

    const shoot = (a, b) => {

      alert(b.type);

          /\*

          'b' represents the React event that triggered the function.

      In this case, the 'click' event

          \*/

    }

    return (

      <button onClick={(event) => shoot("Goal!", event)}>Take the shot!</button>

    );

  }