Class-01

ReactJS

=======

**It is a declarative, efficient and flexible javascript library which is used to develop frontend applications and User interfaces.**

**It is an open source, component based javascript frontend library responsible only for view layer of the application.**

It is developed by **Jordon Walke** the popular software engineer at facebook.

It is initially developed and maintained by the **faceoook** and later it is used in their own products like whatsapp and instagram.

It was realease to the public in the month of **May, 2013.**

The official website of reactjs is http://www.react.dev.

The latest version of reactjs is v18.2.2.

**It is used to create reusable components.**

A component is a building block of react application.

Advantages of ReactJS

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

1) It is easy to learn and easy to use.

2) It supports virtual DOM.

3) It supports one way data binding.

4) Good documentation and community support.

5) It supported by all major browsers.

6) It is used to create reusable components.

Q) What is the difference between Angular and react?

Angular React

----------- ----------

It is a product of Google. It is a product of Facebook.

It is a open source javascript framework It is a javascript frontend library

for web and mobile development. responsible only for view layer of the

application.

It is developed in Oct, 2015. It is developed in May, 2013.

It supports two way data binding. It supports one way data binding.

It uses tradition DOM. It uses virtual DOM.

It is used to develop rich featured It is used to develop Single Page Application

application. (SPA).

Typescript language is used. JSX language is used.

Jasmine and karma is used as a testing Jest and Enzyme is used as a testing

frameworks. frameworks.

It uses default port number i.e 4200. It uses default port number i.e 3000.

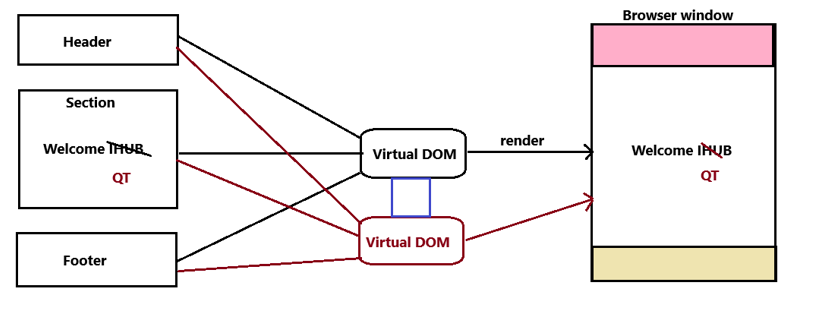
Angular used by Google, Delta, Gmail, React used by Facebook , whatsapp, instagram,

Samsung, Paypal and etc. Airbnb, Netflix and etc.

How ReactJs works internally

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

Diagram: react1.1



React uses a virtual DOM that is basically a DOM tree representation in JavaScript.

So when it needs to read or write to the DOM, it will use the virtual representation of it.Then the virtual DOM will try to find the most efficient way to update the browser's DOM.

Assume we have created multiple components and consistently we are performing some changes in our application.Now we need to see ,how virtual DOM react on each change.

Class-02

JSX

=====

**JSX stands for JavaScript XML.**

**JSX allows us to write HTML in javascript.**

**JSX elements having tag name, attributes and children.**

JSX is not neccessity to develop react applications instead we can use Babel.

**JSX makes our code simpler and elegant.**

**Ultimately JSX transpile to pure javascript which is understandable by the browser.**

JSX elements

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

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

ex:

---

JSX

----

<h1> Heading Tag </h1>

Babel

-----

React.createElement("h1",null,"Heading Tag");

h1 - It is a tag name

null - It is a optional attribute

Heading Tag - It is a children

ex:

---

JSX

----

<div>

<h1> Heading </h1>

</div>

Babel

-----

React.createElement("div",null,React.createElement("h1",null,"Heading"));

ex:

---

JSX

----

<h1 id="myId"> Heading </h1>

Babel

-----

React.createElement("h1",{id:"myId"},"Heading");

ex:

---

JSX

---

<h1 className="myClass">Heading</h1>

Babel

----

React.createElement("h1",{className:"myClass"},"Heading");

ex:

---

JSX

---

<h1 id="myId" className="myClass">Heading</h1>

Babel

----

React.createElement("h1",{id:"myId",className:"myClass"},"Heading");

JSX Expression

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

**JSX allows us to write expressions inside curly braces { }.**

**The expression can be a React variable, or property, or any other valid JavaScript expression.**

**JSX will execute the expression and return the result**.

ex:

1)

let name="Ihub Talent";

<h1>{name}</h1>

2)

<h1>{10\*20}</h1>

3)

<h1>{Math.random()}</h1>

NPM

=====

**NPM stands for Node Package Manager**

**It is a integrated tool for nodejs.**

**npm is used to download or install node dependencies or package or library.**

WE need to below command to download node dependencies.

ex:

npm install -g <dependency\_name>

All the dependencies installed in "node\_modules" folder.

NPM

=====

NPM stands for Node Package Manager.

It is a integrated tool for nodejs.

It is used to download node depedencies or packages.

To downoad node packages and dependencies we need to use below command.

ex:

npm install -g <node\_package\_name>;

All node packages will installed in a "node\_modules" folder.

Steps to setup npm for react applications

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

step1:

------

Download and install nodejs.

ex:

https://nodejs.org/en/blog/release/v16.16.0

step2:

-----

Copy nodejs directory from C drive.

ex:

C:\Program Files\nodejs

step3:

------

Paste nodejs directory in environmental variables.

ex:

Right click to this pc --> properties --> Advanced system settings -->

environment variables --> user variables --> click to new button

variable name : path

variable value : C:\Users\DELL\AppData\Roaming\npm;C:\Program Files\nodejs;

--> ok --> ok --> ok.

step4:

-----

Check the version of nodejs and npm.

ex:

npm -version

node --version

First React application Development

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

step1:

------

Make sure nodejs and npm setup done perfectly.

step2:

------

Download and Install VSC code editor.

ex:

https://code.visualstudio.com/download

step3:

-------

Create a "Reactprojects" folder inside 'E' drive.

step4:

-----

Open the VSC editor from "Reactprojects" folder.

ex:

Reactprojects> code .

step5:

-----

Install node module to create react applications.

ex:

Reactprojects> npm install -g create-react-app

step6:

-----

Create a react project i.e myapp1.

ex:

Reactprojects> npx create-react-app myapp1

step7:

-----

Switch to project i.e myapp1.

ex:

Reactprojects> cd myapp1;

step8:

-----

Run the react application.

ex:

Reactprojects/myapp1> npm start

step9:

-----

Test the react application.

ex:

http://localhost:3000

Class-03

React project structure

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

myapp1

|

|---**node\_modules (it contains all the libararies need react development )**

|

|---public

|

|--- index.html

|--- manifest.json

|--- favicon.ico

|

|------src

|

|--- index.js

|--- index.css

|

|--- App.js

|--- App.css

|--- App.test.js

|

|---package.json

|---README.md

**A "myapp1" is react application name.**

A "node\_modules" contains packages and dependencies installed.

**A "index.html" is a main template of react application**. (it only one html file present in react. This html page only **display in webroser** by taking content from **app.js**

A "manifest.json" file contains metadata which is used when web application installed in

user's computer or laptop or mobile.

A "favicon.ico" is a favourite icon.

A "App.css" file is related to App.js and it is global.

**A "App.js" is a parent component**.

A "App.test.js" file is for unit testing.

A "index.css" file is releated to index.js and it is global.

**A "index.js" is a entry point.**

**A "package.json" file contains dependencies which we used in our project along with their version.**

React Second Application Development

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

project structure

-----------------

myapp2

|

|---node\_modules

|

|---public

|

|--- index.html

|--- manifest.json

|--- favicon.ico

|

|------src

|

|--- index.js

|--- index.css

|

|--- App.js

|--- App.css

|--- App.test.js

|

|---package.json

|---README.md

step1:

------

Create a react application i.e myapp2

ex:

Reactprojects> npx create-react-app myapp2

step2:

-----

Open the VSC editor.

ex:

Reactprojects> code .

step3:

-----

Switch to the project.

ex:

Reactprojects> cd myapp2

step4:

------

Run react application.

ex:

Reactprojects/myapp2> npm start

step5:

------

Test the application by using below request url.

ex:

http://localhost:3000/

step6:

------

Goto App.js file and create custom logic.

App.js

-------

function App()

{

return (

<h1>Welcome to React</h1>

)

}

export default App;

step7:

------

Goto index.js file and perform following changes.

index.js

--------

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App />

<App />

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

**React Fragment**

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

**By default every react component returns only one element at a time.**

**To return more then one element in a react component we need to use react fragment.**

**A React fragment is a feature that allows developers to group multiple elements in a React component without adding extra nodes to the DOM.**

(in three approchaes we can handle is react fragment

**1. <React.Fragment> </React.Fragment>**

**2. <Fragment> </Fragment>**

**3.<> </>**

ex:

---

function App()

{

return (

<div>

<h1>Welcome to React</h1>

<h1>Welcome to IHUB </h1>

</div>

)

}

export default App;

In the above example we are adding extra node to the DOM and it is not recommanded.

ex:

---

import React from 'react';

function App()

{

return (

<React.Fragment>

<h1>Welcome to React</h1>

<h1>Welcome to IHUB </h1>

</React.Fragment>

)

}

export default App;

or

ex:

---

import {Fragment} from 'react';

function App()

{

return (

<Fragment>

<h1>Welcome to React</h1>

<h1>Welcome to IHUB </h1>

<p> Welcome to QT </p>

</Fragment>

)

}

export default App;

or

ex:

---

function App()

{

return (

<>

<h1>Welcome to React</h1>

<h1>Welcome to IHUB </h1>

<p> Welcome to QualityThought </p>

</>

)

}

export default App;

Class-04

React components

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

**Component is a building block of react application**.

**Components allows us to split our UI into reusable independent pieces.**

ex:

<Form>, <Table>, <Button>, <Navbar> and etc.

**Components are javascript functions they accept arbitary inputs like props and return react element to describe how an element should look like.**

**React component must and should starts with uppercase letter.**

There are two ways to create react components.

1) Function Component / Functional Component

2) Class Component

1) Function Component

----------------------

**Function component is also known as stateless component.**

**Function component is a javascript function which takes props as an argument along with input values and returns react elements.**

**There are three ways to create function component in react.**

ex:1

----

function App() (named function)

{

return ()

}

export default App;

ex:2

----

var App=function() (anonomus function)

{

return ()

}

export default App;

ex:3

----

var App=()=> (Array function)

{

return ()

}

export default App;

project structure

------------------

myapp3

|

|---node\_modules

|

|---public

| |

|---index.html

|---manifest.json

|---favicon.ico

|-----src

| |

|---index.js

|

|---App.js

|

|

|---package.json

|---READMD.md

step1:

------

Create a react application i.e myapp3.

ex:

Reactprojects> npx create-react-app myapp3

step2:

------

Open VSC code editor.

ex:

Reactprojects> code .

step3:

------

Switch to the project.

ex:

Reactprojects> cd myapp3

step4:

------

Run the react project.

ex:

Reactproject/myapp3> npm start

step5:

------

Test the application by using below request url.

ex:

http://localhost:3000

step6:

------

Add function component in App.js file.

App.js

-------

var App=()=>{

return(

<h1> Function Component </h1>

)

}

export default App;

2) Class component

------------------

**A class component is also known as statefull component.**

**To create a class component we need to extends a class with react Component class.**

**A class component must implement render() method which returns react elem**ents.

ex:

---

import {Component} from 'react';

class App extends Component

{

render()

{

return (

-

-

)

}

}

export default App;

project structure

------------------

myapp4

|

|---node\_modules

|

|---public

| |

|---index.html

|---manifest.json

|---favicon.ico

|-----src

| |

|---index.js

|

|---App.js

|

|

|---package.json

|---READMD.md

step1:

------

Create a react application i.e myapp4.

ex:

Reactprojects> npx create-react-app myapp4

step2:

------

Open VSC code editor.

ex:

Reactprojects> code .

step3:

------

Switch to the project.

ex:

Reactprojects> cd myapp4

step4:

------

Run the react project.

ex:

Reactproject/myapp4> npm start

step5:

------

Test the application by using below request url.

ex:

http://localhost:3000

step6:

------

Add function component in App.js file.

App.js

-------

import {Component} from 'react';

class App extends Component

{

render()

{

return (

<h1>Class Component </h1>

)

}

}

export default App;

Composing component

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

**If a component refers to another component such concept is called composing component.**

**Ex:**

**(Table.js component are writing in app.js component)**

App.js

------

import Table from "./Table";

function App()

{

return (

<Table/>

)

}

project structure

------------------

myapp5

|

|---node\_modules

|

|---public

| |

|---index.html

|---manifest.json

|---favicon.ico

|-----src

| |

|---index.js

|

|---App.js

|---Table.js (custom component)

|

|---package.json

|---READMD.md

step1:

------

Create a react application i.e myapp5.

ex:

Reactprojects> npx create-react-app myapp5

step2:

------

Open VSC code editor.

ex:

Reactprojects> code .

step3:

------

Switch to the project.

ex:

Reactprojects> cd myapp5

step4:

------

Run the react project.

ex:

Reactproject/myapp5> npm start

step5:

------

Test the application by using below request url.

ex:

http://localhost:3000

step6:

------

Create a Table.js file inside "src" folder.

Table.js

--------

function Table()

{

return (

<>

<table border="1" width="100%">

<thead>

<tr>

<th>SNO</th>

<th>SNAME</th>

<th>SADD</th>

</tr>

</thead>

<tbody>

<tr>

<td>101</td>

<td>Alan</td>

<td>Florida</td>

</tr>

<tr>

<td>102</td>

<td>Jack</td>

<td>Texas</td>

</tr>

<tr>

<td>103</td>

<td>Nelson</td>

<td>Chicago</td>

</tr>

</tbody>

</table>

</>

)

}

export default Table;

step7:

------

Call Table component inside "App.js" file.

App.js

------

import Table from "./Table";

function App()

{

return (

<Table/>

)

}

export default App;

Class-05

Function component with props

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

myapp6

|

|---node\_modules

|

|---public

|

|---index.html

|---manifest.json

|---favicon.ico

|

|------src

|

|---index.js

|---App.js

|

|---package.json

|---README.md

step1:

-----

Create a react project i.e myapp6.

ex:

Reactprojects> npx create-react-app myapp6

step2:

-----

Open the VSC editor.

ex:

Reactprojects> code .

step3:

-----

Switch to the project.

ex:

Reactprojects> cd myapp6

step4:

-----

Run the react application.

ex:

Reactprojects/myapp6> npm start

step5:

----

**Add "ES7 React/Redux/GraphQL" extenion in VSC editor.**

**ex:**

**rfce - Named function component**

**rafce - Arrow function component**

**imr - import React**

**imc - import React and component**

**rcc - class component**

step6:

-----

Create a index.js file and pass some data to App.js file.

index.js

---------

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App name="Alan" rollno="101"/>

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

step7:

-----

Create App.js file and display props values.

App.js

------

import React from 'react'

function App(props)

{

return (

<div>

<h1>Name : {props.name}</h1>

<h1>Roll No : {props.rollno}</h1>

</div>

)

}

export default App

Class component with props

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

In class component we can display props values i.e this.props.

myapp6

|

|---node\_modules

|

|---public

|

|---index.html

|---manifest.json

|---favicon.ico

|

|------src

|

|---index.js

|---App.js

|

|---package.json

|---README.md

step1:

-----

Create a react project i.e myapp6.

ex:

Reactprojects> npx create-react-app myapp6

step2:

-----

Open the VSC editor.

ex:

Reactprojects> code .

step3:

-----

Switch to the project.

ex:

Reactprojects> cd myapp6

step4:

-----

Run the react application.

ex:

Reactprojects/myapp6> npm start

step5:

----

Add "ES7 React/Redux/GraphQL" extenion in VSC editor.

ex:

rfce - Named function component

rafce - Arrow function component

imr - import React

imc - import React and component

rcc - class component

step6:

-----

Create a index.js file and pass some data to App.js file.

index.js

---------

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App name="Alan" rollno="101"/>

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

step7:

-----

Create App.js file and display props values.

App.js

------

import React, { Component } from 'react'

export default class App extends Component {

render() {

return (

<div>

<h1>Name : {this.props.name}</h1>

<h1>Roll No : {this.props.rollno}</h1>

</div>

)

}

}

React CSS

=========

There are two ways to use CSS in react components.

1) Using inline styling

2) Using CSS stylesheet

1) Using inline styling

--------------------------

In react inline styling represented by using "style" attribute.

Project structure

----------------

myapp6

|

|---node\_modules

|

|---public

|

|---index.html

|---manifest.json

|---favicon.ico

|

|------src

|

|---index.js

|---App.js

|

|---package.json

|---README.md

step1:

-----

Create a react project i.e myapp7.

ex:

Reactprojects> npx create-react-app myapp7

step2:

-----

Open the VSC editor.

ex:

Reactprojects> code .

step3:

-----

Switch to the project.

ex:

Reactprojects> cd myapp7

step4:

-----

Run the react application.

ex:

Reactprojects/myapp7> npm start

step5:

------

Add the css code inside "App.js" file.

App.js

------

import React from 'react'

function App() {

return (

<div>

<h1 style={{color:"blue"}}>Heading Tag </h1>

<p style={{color:"red",backgroundColor:"yellow"}}>Paragraph Tag</p>

</div>

)

}

export default App

or

App.js

-------

import React from 'react'

function App() {

const paraStyle={

color:"blue",

backgroundColor:"cyan",

textAlign:"center",

fontSize:"40px"

}

return (

<div>

<h1 style={{color:"blue"}}>Heading Tag </h1>

<p style={paraStyle}>Paragraph Tag</p>

</div>

)

}

export default App

2) Using CSS stylesheet

------------------------

In CSS stylesheet, we will create a sperate .css file and we will import in any perticular when it is required.

Project structure

---------------------

myapp6

|

|---node\_modules

|

|---public

|

|---index.html

|---manifest.json

|---favicon.ico

|

|------src

|

|---index.js

|---App.js

|---App.css

|

|---package.json

|---README.md

step1:

-----

Create a react project i.e myapp7.

ex:

Reactprojects> npx create-react-app myapp7

step2:

-----

Open the VSC editor.

ex:

Reactprojects> code .

step3:

-----

Switch to the project.

ex:

Reactprojects> cd myapp7

step4:

-----

Run the react application.

ex:

Reactprojects/myapp7> npm start

step5:

------

import App.css file inside App.js file and write below code.

App.js

------

import React from 'react'

import './App.css'

function App() {

return (

<div>

<h1>Heading Tag </h1>

<p className='paraStyle'>Paragraph Tag</p>

</div>

)

}

export default App

step6:

-----

Declare css code inside App.css file.

App.css

--------

body

{

background-color: cyan;

}

h1

{

color:brown;

}

.paraStyle

{

text-align: center;

font-size: 50px;

}

import React from 'react'

const Array=[

  {

      title:'KRISHNA',

      img:"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT7lvNvRmxsgBBInSUEaL5CwCXNvR6sfrFiIg&s",

      disc:'at FlowParserMixin.parseBlockOrModuleBlockBody',

  },

  {

      title:'GANESHA',

      img:"https://thumbs.dreamstime.com/z/vinayagar-images-ai-hd-photo-2023-289916691.jpg",

      disc:'at FlowParserMixin.parseBlockOrModuleBlockBody',

  },

  {

      title:'SHIVA',

      img:"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTpL9bt8-t6xEpd21HJNYsVkTY\_Ygtc2FeACw&s",

      disc:'at FlowParserMixin.parseBlockOrModuleBlockBody',

  }]

const App = () => {

  return (

    <div className='movie-grid'>

      {

        Array.map((Array1)=>(<Movie title={Array1.title} img={Array1.img} disc={Array1.disc}/>

        ))

      }

      </div>

  )

}

const Movie = (props) => {

  return (

    <div className='movie'>

      <h3 style={{textAlign:'center'}}>{props.title}</h3>

      <img src={props.img} alt='iam sorry'/>

      <p>{props.disc}</p>

    </div>

  )

}

export default App

State

======

State is similar to props but is private and fully controlled by a component.

State we can declare only in class component.

State can be modified or updatable where as props is read only.

There are two ways to initialized the state in react.

1) Inside the class

2) Inside the constructor

1) Inside the class

--------------------

Project structure

---------------------

myapp8

|

|---node\_modules

|

|---public

|

|---index.html

|---manifest.json

|---favicon.ico

|

|------src

|

|---index.js

|---App.js

|---App.css

|

|---package.json

|---README.md

step1:

-----

Create a react project i.e myapp8.

ex:

Reactprojects> npx create-react-app myapp8

step2:

-----

Open the VSC editor.

ex:

Reactprojects> code .

step3:

-----

Switch to the project.

ex:

Reactprojects> cd myapp8

step4:

-----

Run the react application.

ex:

Reactprojects/myapp8> npm start

step5:

------

Declare the state inside class component of App.js file.

App.js

-----

import React, { Component } from 'react'

export default class App extends Component

{

state={

rollno:501,

name:"Jose"

}

render()

{

return (

<div>

<h1>Roll No : {this.state.rollno}</h1>

<h1>Name : {this.state.name}</h1>

</div>

)

}

}

2) Inside the constructor

-----------------------

Project structure

---------------------

myapp8

|

|---node\_modules

|

|---public

|

|---index.html

|---manifest.json

|---favicon.ico

|

|------src

|

|---index.js

|---App.js

|---App.css

|

|---package.json

|---README.md

step1:

-----

Create a react project i.e myapp8.

ex:

Reactprojects> npx create-react-app myapp8

step2:

-----

Open the VSC editor.

ex:

Reactprojects> code .

step3:

-----

Switch to the project.

ex:

Reactprojects> cd myapp8

step4:

-----

Run the react application.

ex:

Reactprojects/myapp8> npm start

step5:

------

Declare the state inside class component of App.js file.

App.js

-----

import React, { Component } from 'react'

export default class App extends Component

{

constructor()

{

super();

this.state={

rollno:701,

name:"Jacky"

}

}

render()

{

return (

<div>

<h1>Roll No :{this.state.rollno}</h1>

<h1>Name : {this.state.name}</h1>

</div>

)

}

}

Class-06

Event Handling in React

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

**Action to which javascript response is called event handling.**

**Handling events on react elements is similar to handling events on DOM elements.**

Ex:

JavaScript

-----------

<button onclick="f1()"> clickme </button>

ReactJS

--------

**Function component : <button onClick={handleClick}> clickme </button>**

**Class component : <button onClick={this.handleClick}> clickme </button>**

 **onClick**: Triggered when a user clicks an element.

 **onChange**: Fired when the value of an input, select, or textarea changes.

 **onSubmit**: Fired when a form is submitted.

 **onFocus**: Triggered when an element gains focus.

 **onBlur**: Fired when an element loses focus.

 **onKeyDown**: Triggered when a key is pressed down on an element.

 **onKeyPress**: Fired when a key is pressed and released on an element.

 **onKeyUp**: Triggered when a key is released on an element.

 **onMouseDown**: Fired when a mouse button is pressed on an element.

 **onMouseUp**: Triggered when a mouse button is released on an element.

 **onMouseEnter**: Fired when the mouse pointer enters an element.

 **onMouseLeave**: Triggered when the mouse pointer leaves an element.

 **onMouseMove**: Fired when the mouse pointer moves over an element.

 **onTouchStart**: Triggered when a touch event starts on a device screen.

 **onTouchEnd**: Fired when a touch event ends on a device screen.

 **onTouchMove**: Fired when a touch event moves across a device screen.

 **onWheel**: Triggered when the mouse wheel is used on an element.

 **onDrag**: Fired when an element is being dragged.

 **onDragStart**: Triggered when a drag operation starts on an element.

 **onDragEnd**: Fired when a drag operation ends on an element.

 **onDragOver**: Fired when an element is being dragged over.

 **onDragEnter**: Triggered when the dragged element enters a valid drop target.

 **onDragLeave**: Fired when the dragged element leaves a drop target.

 **onDrop**: Triggered when an element is dropped onto a drop target.

 **onSelect**: Fired when a user selects text inside an input or textarea.

 **onScroll**: Triggered when an element is scrolled.

 **onError**: Fired when an error occurs during the loading of a resource (e.g., an image).

 **onLoad**: Triggered when a resource has finished loading (e.g., image or iframe).

 **onResize**: Fired when the window or an element is resized.

 **onContextMenu**: Triggered when a right-click context menu is opene

Event Handling using function component

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

myapp9

|

|---node\_modules

|

|---public

| |

|---index.html

|---manifest.json

|---favicon.ico

|----src

| |

|---index.js

|---App.js

|

|---package.json

|---READMD.md

step1:

-----

Create react application i.e myapp9.

ex:

Reactprojects> npx create-react-app myapp9

step2:

------

Open VSC editor.

ex:

Reactprojects> code .

step3:

------

Switch to the react project.

ex:

Reactprojects> cd myapp9

step4:

-----

Run the react application.

ex:

Reactprojects/myapp9> npm start

step5:

-----

Write a below code inside App.js file.

App.js

-------

import React from 'react'

function App() {

const handleClick=()=>

{

console.log("You have clicked")

}

return (

<div>

<button onClick={handleClick}> click Me </button>

</div>

)

}

export default App

or

App.js

-------

import React from 'react'

function App() {

const handleClick=(e)=>

{

e.preventDefault();

}

return (

<div>

<a href="http://www.google.com" onClick={handleClick}> click Me </a>

</div>

)

}

export default App

==============chat gpt examples===============

1. Basic console.log Example

This is a simple example where clicking a button logs a message to the console.

javascript

Copy code

import React from 'react';

function App() {

const handleClick = () => {

console.log("Button clicked!");

};

return (

<div>

<button onClick={handleClick}>Click me</button>

</div>

);

}

export default App;

2. Button Toggle (State Change)

This example toggles the button's text and a state value when clicked.

javascript

Copy code

import React, { useState } from 'react';

function App() {

const [isClicked, setIsClicked] = useState(false);

const handleClick = () => {

setIsClicked(!isClicked); // Toggle the state

};

return (

<div>

<button onClick={handleClick}>

{isClicked ? "Clicked!" : "Click me"}

</button>

</div>

);

}

export default App;

3. Prevent Default Behavior

Sometimes, you may want to prevent the default behavior (e.g., form submission or link navigation).

javascript

Copy code

import React from 'react';

function App() {

const handleClick = (e) => {

e.preventDefault(); // Prevent the default action (e.g., form submission)

console.log("Default behavior prevented!");

};

return (

<div>

<form onSubmit={handleClick}>

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

</form>

</div>

);

}

export default App;

4. Passing Parameters to onClick

You can pass arguments or parameters to the onClick handler by using an arrow function.

javascript

Copy code

import React from 'react';

function App() {

const handleClick = (message) => {

console.log(message);

};

return (

<div>

<button onClick={() => handleClick("Hello from the button!")}>

Click me

</button>

</div>

);

}

export default App;

5. Calling Multiple Functions on Click

You can call multiple functions inside an onClick handler.

javascript

Copy code

import React from 'react';

function App() {

const firstFunction = () => {

console.log("First function called!");

};

const secondFunction = () => {

console.log("Second function called!");

};

const handleClick = () => {

firstFunction();

secondFunction();

};

return (

<div>

<button onClick={handleClick}>Click me</button>

</div>

);

}

export default App;

6. Navigating to Another Page (Using window.location)

This example demonstrates how to redirect to another page using the window.location object.

javascript

Copy code

import React from 'react';

function App() {

const handleClick = () => {

window.location.href = "https://www.google.com"; // Redirect to Google

};

return (

<div>

<button onClick={handleClick}>Go to Google</button>

</div>

);

}

export default App;

7. Using setTimeout for Delayed Action

This example uses setTimeout to trigger an action after a delay.

javascript

Copy code

import React from 'react';

function App() {

const handleClick = () => {

setTimeout(() => {

console.log("This was delayed by 2 seconds");

}, 2000);

};

return (

<div>

<button onClick={handleClick}>Click me (Delayed)</button>

</div>

);

}

export default App;

8. Calling a Function with Event Object

You can access the event object directly inside the onClick handler.

javascript

Copy code

import React from 'react';

function App() {

const handleClick = (e) => {

console.log("Button clicked!", e);

e.preventDefault(); // Optionally prevent the default action

};

return (

<div>

<button onClick={handleClick}>Click me</button>

</div>

);

}

export default App;

9. Using window.alert in onClick

Using the browser's alert to show a pop-up when the button is clicked.

javascript

Copy code

import React from 'react';

function App() {

const handleClick = () => {

window.alert("Button was clicked!");

};

return (

<div>

<button onClick={handleClick}>Click me</button>

</div>

);

}

export default App;

10. Handling Form Submission with onClick

Handle form submission using a button click event.

javascript

Copy code

import React, { useState } from 'react';

function App() {

const [inputValue, setInputValue] = useState("");

const handleClick = (e) => {

e.preventDefault(); // Prevent form submission

console.log("Form submitted with value:", inputValue);

};

return (

<div>

<form>

<input

type="text"

value={inputValue}

onChange={(e) => setInputValue(e.target.value)}

/>

<button type="submit" onClick={handleClick}>Submit</button>

</form>

</div>

);

}

export default App;

11. Changing Styles on Click

Change the style of an element dynamically when clicked.

javascript

Copy code

import React, { useState } from 'react';

function App() {

const [color, setColor] = useState('blue');

const handleClick = () => {

setColor(color === 'blue' ? 'red' : 'blue'); // Toggle between blue and red

};

return (

<div>

<button onClick={handleClick} style={{ backgroundColor: color }}>

Click me to change color

</button>

</div>

);

}

export default App;

12. Accessing DOM Elements (Refs) with onClick

You can use refs to directly access DOM elements when a button is clicked.

javascript

Copy code

import React, { useRef } from 'react';

function App() {

const buttonRef = useRef(null);

const handleClick = () => {

buttonRef.current.style.backgroundColor = 'yellow'; // Change button color

};

return (

<div>

<button ref={buttonRef} onClick={handleClick}>Click me to change color</button>

</div>

);

}

export default App;

13. Using useEffect with onClick

Here is an example that shows how useEffect can be triggered after clicking a button.

javascript

Copy code

import React, { useState, useEffect } from 'react';

function App() {

const [clicked, setClicked] = useState(false);

useEffect(() => {

if (clicked) {

console.log("Button clicked, effect triggered!");

}

}, [clicked]);

const handleClick = () => {

setClicked(true);

};

return (

<div>

<button onClick={handleClick}>Click me to trigger useEffect</button>

</div>

);

}

export default App;

Event Handling using class component

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

myapp10

|

|---node\_modules

|

|---public

| |

|---index.html

|---manifest.json

|---favicon.ico

|----src

| |

|---index.js

|---App.js

|

|---package.json

|---READMD.md

step1:

-----

Open the VSC editor.

ex:

Reactprojects> code .

step2:

------

Create a react application i.e myapp10 using VSC terminal.

ex:

npx create-react-app myapp10

step3:

-----

Switch to the project.

ex:

cd myapp10

step4:

------

Run the react application.

ex:

npm start

step5:

-----

Write a below code inside App.js file.

App.js

-------

import React, { Component } from 'react'

export default class App extends Component {

handleClick=()=>

{

console.log("You have clicked once again")

}

render() {

return (

<div>

<button onClick={this.handleClick}> click Me </button>

</div>

)

}

}

React application to update the state

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

In order to update the state in react we need to **use setState() method.**

myapp11

|

|---node\_modules

|

|---public

| |

|---index.html

|---manifest.json

|---favicon.ico

|----src

| |

|---index.js

|---App.js

|

|---package.json

|---READMD.md

step1:

-----

Open the VSC editor.

ex:

Reactprojects> code .

step2:

------

Create a react application i.e myapp10 using VSC terminal.

ex:

npx create-react-app myapp11

step3:

-----

Switch to the project.

ex:

cd myapp11

step4:

------

Run the react application.

ex:

npm start

step5:

-----

Write a below code inside App.js file.

App.js

------

import React, { Component } from 'react'

export default class App extends Component {

state={

name:"Alan",

rollno:101

}

handleClick=()=>

{

this.setState({"name":"Jose","rollno":201});

}

render() {

return (

<div>

<h1>Name : {this.state.name}</h1>

<h1>Roll No : {this.state.rollno}</h1>

<button onClick={this.handleClick}> Update </button>

</div>

)

}

}

**Q) What is the difference between props and state?**

**props state**

**-------------- -------------**

**It is read-only. It is updatable.**

**It is immutable. It is mutable.**

**Stateless components can have props. Statefull components can have state.**

**It allows us to pass the data from It holds information of a component.**

**one component to another component as an**

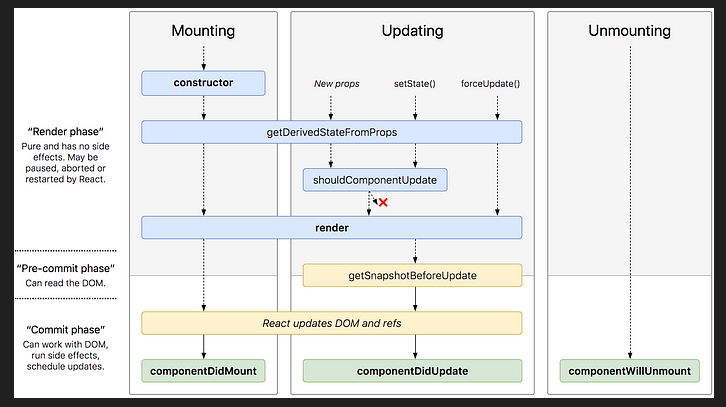
**argument.**

**Child components can access props. It is private so child components can't access.**

**Phases of react components**

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

**Diagram: react5.1**

****

**There are four Phases of components in ReactJS.**

**1)Mounting**

**2)Updating**

**3)Error Handling**

**4)Unmounting**

**1)Mounting**

**-----------**

**Mounting is a process of creating an element and inserting it in a DOM tree.**

**(mounting means creating a node or mermory in the dom)**

**2)Updating**

**-------------**

**Updating is a process of changing state or props of a component and update changes to nodes already existing in the DOM.**

**(mounting means modifying a node or mermory in the dom)**

**3)Error Handling**

**----------------**

**Error Handling used when there is a error during rendering ,in lifecycle**

**method or in the constructor of any child component.**

**4)Unmounting**

**---------------**

**Unmounting is a process of removing elements from the DOM.**

**In general it will clear the reserved memory.**

**(mounting means deleting a node or mermory in the dom)**

**Q)Explain life cycle methods of mounting ?**

**Mounting phase contains four methods.**

**1) constructor()**

**2) getDerivedStateFromProps()**

**3) render()**

**4) componentDidMount()**

**Q)Explain life cycle methods of unmounting?**

**Unmounting phase contains one method.**

**1) componentWillUnmount()**

**Q)Explain life cycle methods of updating?**

**updating phase contains five methods.**

**1) getDerivedStateFromProps**

**2) shouldComponentUpdate()**

**3) render()**

**4) getSnapshotBeforeUpdate()**

**4) ComponentDidUpdate()**

**Class-07**

**Q) What is React Hook?**

**Hooks are new addition in React 16.X version.**

**Hooks we can declare only in function component.**

**Hooks allow function components to have access to state , lifecycle methods and other React features.**

**React provides a few built-In hooks like useState(),useEffect(),useReducer() and etc.**

**It is possible to create custom hook in react.**

**1) useState() hook**

**----------------**

**A useState() is a Hook that allows us to add React state to function components.**

Project structure

-----------------

myapp12

|

|---node\_modules

|

|---public

| |

|---index.html

|---manifest.json

|---favicon.ico

|-----src

| |

|---index.js

|---App.js

|

|---package.json

|---READMD.md

step1:

------

Create a react application i.e myapp12

ex:

npx create-react-app myapp12

step2:

-----

Switch to the project.

ex:

cd myapp12

step3:

------

Run react application.

ex:

npm start

step4:

------

Test the application by using below request url.

ex:

http://localhost:3000

step5:

-----

Write a below code inside App.js file.

App.js

------

import React from 'react'

import { useState } from 'react'

function App()

{

const [name,setName]=useState("Alan");

const handleClick=()=>

{

setName("Jose");

}

return (

<div>

<h1>Name : {name} </h1>

<button onClick={handleClick}> Change </button>

</div>

)

}

export default App

**2) useEffect() hook**

**-------------------**

**It is used to perform side effects in function components.**

**Data fetching , setting up a subscription, changing the document title and manually changing the DOM in React components are examples of side effects.**

Project structure

-----------------

myapp12

|

|---node\_modules

|

|---public

| |

|---index.html

|---manifest.json

|---favicon.ico

|-----src

| |

|---index.js

|---App.js

|

|---package.json

|---READMD.md

step1:

------

Create a react application i.e myapp12

ex:

npx create-react-app myapp12

step2:

-----

Switch to the project.

ex:

cd myapp12

step3:

------

Run react application.

ex:

npm start

step4:

------

Test the application by using below request url.

ex:

http://localhost:3000

step5:

-----

Write a below code inside App.js file.

App.js

------

import React from 'react'

import { useState,useEffect } from 'react';

function App()

{

const [count,setCount]=useState(0);

const handleClick=()=>

{

setCount(count+1);

}

useEffect(()=>{

document.title=`You have ${count} clicked`;

});

return (

<div>

<h1>Count : {count} </h1>

<button onClick={handleClick}> Increment </button>

</div>

)

}

export default App

**3) useReducer() hook**

**--------------------**

**The useReducer() Hook is similar to the useState() Hook.**

**It allows for custom state logic.**

**The useReducer Hook accepts two arguments.**

**ex:**

**useReducer(<reducer>, <initialState>)**

**The useReducer Hook returns the current state and a dispatch method.**

**ex:**

**const [state,dispatch]= userReducer(reducer,initialState);**

Project structure

-----------------

myapp12

|

|---node\_modules

|

|---public

| |

|---index.html

|---manifest.json

|---favicon.ico

|-----src

| |

|---index.js

|---App.js

|

|---package.json

|---READMD.md

step1:

------

Create a react application i.e myapp12

ex:

npx create-react-app myapp12

step2:

-----

Switch to the project.

ex:

cd myapp12

step3:

------

Run react application.

ex:

npm start

step4:

------

Test the application by using below request url.

ex:

http://localhost:3000

step5:

-----

Write a below code inside App.js file.

App.js

----------

import {useReducer} from 'react';

const initialState=0;

const reducer=(state,action)=>

{

switch(action)

{

case 'increment': return state+1;

case 'decrement': return state-1;

case 'reset': return initialState;

default: return state;

}

}

function App()

{

const [count,dispatch]=useReducer(reducer,initialState);

return (

<div>

<h1>Count : {count}</h1>

<button onClick={()=>dispatch('increment')}>Increment</button>

<button onClick={()=>dispatch('decrement')}>Decrement</button>

<button onClick={()=>dispatch('reset')}>Reset</button>

</div>

)

}

export default App;

Class-08

**Custom Hooks**

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

**Hooks which are created by the user based on the application requirement are called custom hooks.**

ex:

myCustomHook()

customHook()

ihubHook()

myCustomCounter()

Project Structure

-------------------

myapp13

|

|----node\_modules

|

|----public

| |

|----favicon.ico

|----index.html

|----manifest.json

|

|-----src

| |

|----index.js

|----App.js

|----CustomHook.js

|

|-----package.json

|-----README.md

CustomHook.js

---------------

import React from 'react'

import {useState} from 'react'

function CustomHook()

{

const [count,setCount]=useState(0);

const handleClick=()=>

{

setCount(count+1);

}

return(

{

count,

handleClick

})

}

export default CustomHook

App.js

--------

import React from 'react'

import customHook from './CustomHook';

function App() {

const data=customHook();

return (

<div>

<h1>Count : {data.count}</h1>

<button onClick={data.handleClick}>Increment</button>

</div>

)

}

export default App

**Assets/Images in ReactJS**

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

**We can declare react assets/images in two places.**

**1) Inside public folder**

**2) Inside src folder**

2) Inside src folder

--------------------

myapp14

|

|----node\_modules

|

|----public

| |

|---favicon.ico

|---manifest.json

|---index.html

|-----src

|

|---index.js

|---App.js

|

|---images

|

|---thumbnail1.jpg

|---thumbnail2.jpg

|---thumbnail3.jpg

|

|-----package.json

|-----README.md

step1:

-----

Create a react project i.e myapp14.

ex:

reactprojects> npx create-react-app myapp14

step2:

------

Create a image folder inside "src".

step3:

-----

Add the images inside "src/images" folder.

step4:

------

Goto App.js and remove the code.

step5:

------

Add the below code in App.js file.

App.js

------

import pic1 from './images/thumbnail1.jpg';

import pic2 from './images/thumbnail2.jpg';

import pic3 from './images/thumbnail3.jpg';

function App()

{

const imgStyle={

width:"300px",

height:"300px"

}

return (

<>

<img src={pic1} style={imgStyle}/>

<img src={pic2} style={imgStyle}/>

<img src={pic3} style={imgStyle}/>

</>

)

}

export default App;

step6:

-----

Switch to react project.

ex:

reactprojects> cd myapp12

step7:

-----

Run the react project.

ex:

reactprojects/myapp12> npm start

step8:

------

Test the react application by using below request url.

ex:

http://localhost:3000/

**React Router**

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

**Routing is a process in which a user is directed to different pages based on their**

**action or request.**

**ReactJS Router is mainly used for developing Single Page Web Applications.**

**React Router is used to define multiple routes in the application.**

**When a user types a specific URL into the browser, and if this URL path matches any**

**'route' inside the router file, the user will be redirected to that particular route.**

**React Router is a standard library system built on top of the React and used to**

**create routing in the React application using React Router Package.**

**React contains three different packages for routing.**

**1)react-router:**

**----------------**

**It provides the core routing components and functions for the React Router applications.**

**2)react-router-native:**

**--------------------**

**It is used for mobile applications.**

**3)react-router-dom:**

**-------------------**

**It is used for web applications design.**

**Note:**

**-------**

**It is not possible to install react-router directly in your application.**

**To use react routing, first, you need to install react-router-dom modules in your application.**

**We have two types of router components.**

**1)<BrowserRouter>:**

**-------------------**

**It is used for handling the dynamic URL.**

**2)<HashRouter>:**

**--------------**

**It is used for handling the static request.**

Project structure

--------------------

myapp15

|

|------node-modules

|

|------public

| |

| |------favicon.ico

| |------index.html

| |------manifest.json

|

|------src

|

|------index.js

|------App.js (Routing File)

|------Home.js

|------About.js

|------Contact.js

|------Error.js

|

|------package.json

|------README.md

step1:

------

create react "myapp15" project in VSC.

ex:

projects>create-react-app myapp15

step2:

--------

Move to myapp15 project.

ex:

project> cd myapp15

step3:

------

install react router dom.

ex:

project/myapp15>npm install --save react-router-dom

step4:

-------

Restart the application .

ex:

myapp15> npm start

step5:

--------

create App.js,Home.js,About.js ,Contact.js and Error.js component inside "src" folder.

App.js

-------

import Home from './Home';

import Contact from './Contact';

import About from './About';

import Error from './Error'

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

function App() {

return (

<div>

<BrowserRouter>

<Routes>

<Route exact path="/" element={<Home />}/>

<Route path="/about" element={<About />}/>

<Route path="/contact" element={<Contact />}/>

<Route path="\*" element={<Error />}/>

</Routes>

</BrowserRouter>

</div>

);

}

export default App;

Home.js

----------

function Home()

{

return (

<div>

<h1>Home</h1>

</div>

)

}

export default Home;

About.js

---------

function About()

{

return (

<div>

<h1>About</h1>

</div>

)

}

export default About;

Contact.js

----------

function Contact()

{

return (

<div>

<h1>Contact</h1>

</div>

)

}

export default Contact;

Error.js

----------

function Error()

{

return(

<div>

<h1>OOPS! 404 Error </h1>

</div>

)

}

export default Error;

step6:

------

create index.js component to render the output inside "src" folder.

index.js

---------

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App/>

</React.StrictMode>

);

step7:

-------

Test the application by using below url's.

ex:

http://localhost:3000/

http://localhost:3000/home

http://localhost:3000/about

http://localhost:3000/contact

http://localhost:3000/gallery

http://localhost:3000/services

**Adding Navigation using Link component**

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

**A Link component is used to create links which allow to navigate on different URLs and render its content without reloading the webpage.**

ex:2

-------

App.js

---------

import Home from './Home';

import Contact from './Contact';

import About from './About';

import Error from './Error'

import {Link, Routes,Route,BrowserRouter } from 'react-router-dom'

function App() {

return (

<div>

<BrowserRouter>

<nav >

<Link style={{display:"block"}} to="/">Home</Link>

<Link style={{display:"block"}} to="/about">About Us</Link>

<Link style={{display:"block"}} to="/contact">Contact US</Link>

</nav>

<Routes>

<Route exact path="/" element={<Home />}/>

<Route path="/about" element={<About />}/>

<Route path="/contact" element={<Contact />}/>

<Route path="\*" element={<Error />}/>

</Routes>

</BrowserRouter>

</div>

);

}

export default App;

Home.js

-----------

function Home()

{

return (

<div>

<h1>Home</h1>

</div>

)

}

export default Home;

About.js

------------

function About()

{

return (

<div>

<h1>About</h1>

</div>

)

}

export default About;

Contact.js

--------------

function Contact()

{

return (

<div>

<h1>Contact</h1>

</div>

)

}

export default Contact;

Error.js

----------

function Error()

{

return(

<div>

<h1>OOPS! 404 Error </h1>

</div>

)

}

export default Error;

index.js

-----------

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App/>

</React.StrictMode>

);

**Q) Difference between function component vs class component?**

**function component class component**

**------------------ ----------------**

**It is also known as stateless It is a statefull component.**

**component.**

**In a function component we will use In a class component we will use**

**return keyword. render() method.**

**It supports hooks. It does not support hooks.**

**Constructor is not used. Constructor is used.**

**ex: ex:**

**import {Component} from 'react';**

**function App export default class App extends Component**

**{ {**

**return ( render()**

**<h1> Hello </h1> {**

**) return (**

**} <h1>Hello</h1>**

**export default App; )**

**}**

**}**

**Q)Difference between real dom vs virtual dom ?**

**Real dom virtual dom**

**----------- --------------**

**It updates slow. It updates faster.**

**Can directly updates HTML. Can't directly updates HTML.**

**Creates a new dom if element updates. Update the jsx if element updates.**

**DOM manipulation is very expensive. DOM manipulation is very easy.**

**Too much of memory wastage. No memory wastage.**

**Class-09**

**Bootstrap in React**

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

**A Single-page applications gaining popularity over the last few years, so many front-end frameworks have introduced such as Angular, Vue, Ember, etc. As a result, jQuery is not a necessary requirement for building web apps.**

**Currently, React is mostly used JavaScript library for building web applications, and Bootstrap become the most popular CSS framework.**

**Let see how to use bootstrap in react applications.**

Project structure

-------------------

myapp16

|

|----node\_modules

|

|----public

|

|---favicon.ico

|---index.html

|---manifest.json

|

|------src

| |

|---index.js

|

| |---App.js

|

|------package.json

|

|------README.md

step1:

-----

create a react project i.e myapp16

ex:

Reactprojects> npx create-react-app myapp16

step2:

------

Open the VSC code editor.

ex:

Reactprojects> code .

step3:

-----

Move/Switch to myapp16 project.

ex:

Reactprojects> cd myapp16

step4:

-------

Install Bootstrap package.

ex:

Reactprojects/myapp16> npm install bootstrap

step5:

-------

Run the react application.

ex:

Reactprojecs/myapp16> npm start

step6:

-----

Create a App.js file inside "src" folder.

App.js

-------

function App()

{

return(

<div className="container mt-5">

<button className="btn btn-outline-primary">clickMe</button>

</div>

)

}

export default App;

step7:

------

Import bootstrap package inside "index.js" file.

index.js

---------

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import '../node\_modules/bootstrap/dist/css/bootstrap.css';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

reportWebVitals();

step8:

------

Test the application by using below request url.

ex:

http://localhost:3000

**React Forms**

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

**Forms are an integral part of any modern web application.**

**It allows the users to interact with the application as well as gather information from the users.**

**Forms can perform many tasks that depend on the nature of your business requirements and logic such as authentication of the user, adding user, searching, filtering, booking, ordering, etc.**

**React offers a stateful, reactive approach to build a form.**

**The component rather than the DOM usually handles the React form.**

**In React, the form is usually implemented by using controlled components.**

Project structure

------------------

myapp17

|

|----node\_modules

|

|----public

|

|---favicon.ico

|---index.html

|---manifest.json

|------src

|

|---index.js

|

|---App.js

|

|-------package.json

|

|-------README.md

step1:

------

create a react project i.e myapp17.

ex:

Reactprojects> npx create-react-app myapp17

step2:

-------

Open the VSC code Editor.

ex:

Reactprojects> code .

step3:

-----

Switch/Move to myapp14 project.

ex:

Reactprojects> cd myapp17

step4:

------

Install bootstrap package.

ex:

Reactprojects/myapp17> npm install bootstrap

step5:

------

Run the react application.

ex:

Reactprojects/myapp17> npm start

step6:

-------

Import Bootstrap package inside "index.js" file.

index.js

---------

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import '../node\_modules/bootstrap/dist/css/bootstrap.css';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

step7:

------

Create App.js file inside "src" folder.

App.js

-----

import {useState} from 'react';

function App()

{

const [userRegistration,setUserRegistration]=useState({

username:"",

password:"",

date:"",

category:""

})

const handleClick=(e)=>

{

const name=e.target.name;

const value=e.target.value;

//set to state

setUserRegistration({... userRegistration,[name]:value});

}

const handleSubmit=(e)=>

{

e.preventDefault();

setUserRegistration({username:"",password:"",date:"",category:""});

}

return(

<div className="container mt-4">

<form onSubmit={handleSubmit}>

<div className="row w-50">

<h1 className="text-center" ><u>React Form </u></h1>

<label htmlFor="username" className="my-3">UserName:</label>

<input type="text" name="username" autocomplete="off"

className="form-control"

value={userRegistration.username}

onChange={handleClick}/>

<label htmlFor="password" className="my-3">Password:</label>

<input type="password" name="password" autocomplete="off"

className="form-control"

value={userRegistration.password}

onChange={handleClick}/>

<label htmlFor="date" className="my-3">Date:</label>

<input type="date" name="date" autocomplete="off"

className="form-control"

value={userRegistration.date}

onChange={handleClick}/>

<label htmlFor="category" className="my-3">Category</label>

<select name="category" className="form-control"

value={userRegistration.category}

onChange={handleClick}>

<option value="">none</option>

<option value="entertainment">Entertainment</option>

<option value="drama">Drama</option>

<option value="action">Action</option>

</select>

<button className="btn btn-primary mt-4 w-100"> submit </button>

</div>

</form>

</div>

)

}

export default App;

step8:

------

Test the application by using below request url.

ex:

http://localhost:3000

**Lists in ReactJs**

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

**Lists are used to display data in an ordered format and mainly used to**

**display menus on websites. In React, Lists can be created in a similar way as we create lists in JavaScript. Let us see how we transform Lists in regular JavaScript.**

**The map() function is used for traversing the lists.**

ex:

Project structure

-----------------

myapp18

|

|----node\_modules

|

|-----public

| |

|---favicon.ico

|---index.html

|---manifest.json

|

|-----src

|

|---index.js

|---App.js

|

|-----package.json

|-----README.md

step1:

-----

create a react project i.e myapp18.

ex:

Reactprojects> npx create-react-app myapp18

step2:

------

Open the VSC code editor.

ex:

Reactprojects> code .

step3:

-----

Move/Switch to myapp18 project.

ex:

Reactprojects> cd myapp18

step4:

-----

Run the react application.

ex;

Reactprojects/myapp18> npm start

step5:

------

Create App.js file inside "src" folder.

App.js

-------

import React, { Component } from 'react'

export default class App extends Component {

render() {

var arr=[10,20,30,40];

var newArr=arr.map((element)=>

{

return <li>{element}</li>

})

return (

<ul>

{newArr}

</ul>

)

}

}

step6:

-----

Test the application by using below request url.

ex:

http://localhost:3000

Class-10

ex:2

-----

App.js

------

import React, { Component } from 'react'

export default class App extends Component {

state={

users:[

{pid:101,pname:"LG",pprice:10000},

{pid:102,pname:"LAVA",pprice:20000},

{pid:103,pname:"MI",pprice:30000},

{pid:104,pname:"SAMSUNG",pprice:40000}

]

}

render() {

var newArr=this.state.users.map(user=>

{

return <h1>Id: {user.pid} Name: {user.pname} Price: {user.pprice}</h1>

})

return (

<div>

{newArr}

</div>

)

}

}

ex:3

-----

App.js

-------

import React, { Component } from 'react'

export default class App extends Component {

state={

users:[

{pid:101,pname:"LG",pprice:10000},

{pid:102,pname:"LAVA",pprice:20000},

{pid:103,pname:"MI",pprice:30000},

{pid:104,pname:"SAMSUNG",pprice:40000}

]

}

render() {

var newArr=this.state.users.map(user=>

{

return <tr><td>{user.pid}</td> <td> {user.pname}</td> <td>{user.pprice}</td></tr>

})

return (

<div>

<table border={1} width="100%">

<thead>

<tr>

<th>ID</th>

<th>NAME</th>

<th>PRICE</th>

</tr>

</thead>

<tbody>

{newArr}

</tbody>

</table>

</div>

)

}

}

**Key in ReactJS**

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

**A key is a special string attribute you need to include when creating**

**lists of elements.**

**Keys help react identify which items have changed are added or are removed.**

ex:

App.js

-----

import React, { Component } from 'react'

export default class App extends Component {

state={

users:[

{pid:101,pname:"LG",pprice:10000},

{pid:102,pname:"LAVA",pprice:20000},

{pid:103,pname:"MI",pprice:30000},

{pid:104,pname:"SAMSUNG",pprice:40000}

]

}

render() {

var newArr=this.state.users.map(user=>

{

return <tr key={user.pid}><td>{user.pid}</td> <td> {user.pname}</td> <td>{user.pprice}</td></tr>

})

return (

<table border={1} width="100%">

<thead>

<tr>

<th>ID</th>

<th>NAME</th>

<th>PRICE</th>

</tr>

</thead>

<tbody>

{newArr}

</tbody>

</table>

)

}

}

**Axios**

**=======**

**Axios is used to make HTTP request (GET,POST,PUT,DELETE).**

**Using axios we can give the request to Rest API's.**

**We can install axios by using below command.**

ex:

reactprojects> npm install axios

or

reactprojects> yarn add axios

Project structure

-----------------

myapp19

|

|----node\_modules

|

|-----public

| |

|---favicon.ico

|---index.html

|---manifest.json

|

|-----src

|

|---index.js

|---App.js

|---FetchApi.js

|

|-----package.json

|-----README.md

step1:

-----

create a react project i.e myapp19.

ex:

Reactprojects> npx create-react-app myapp16

step2:

------

Open the VSC code editor.

ex:

Reactprojects> code .

step3:

-----

Move/Switch to myapp19 project.

ex:

Reactprojects> cd myapp19

step4:

-----

Install axios in myapp19 project.

ex:

Reactprojects/myapp19> npm install axios

step5:

-------

Run the react application.

ex;

Reactprojects/myapp19> npm start

step6:

------

Create App.js file inside "src" folder.

App.js

------

import FetchApi from "./FetchApi";

function App()

{

return (

<FetchApi/>

)

}

export default App;

step7:

-------

Arange one REST API for fetching the data.

ex:

https://jsonplaceholder.typicode.com/users

step8:

-------

Create FetchApi.js file inside "src" folder.

FetchApi.js

-----------

import {useState} from 'react';

import axios from 'axios';

function FetchApi()

{

const [data,setData]=useState([])

const handleClick=()=>

{

axios.get("https://jsonplaceholder.typicode.com/users")

.then(response=>

{

setData(response.data)

})

.catch(error=>

{

this.setData(error);

})

}

return (

<div>

<center>

<button onClick={handleClick}>Fetch API </button>

</center>

<table border={1} width="100%">

<thead>

<tr>

<th>ID</th>

<th>NAME</th>

<th>USERNAME</th>

<th>EMAIL</th>

</tr>

</thead>

<tbody>

{

data.map(data=>

{

return <tr>

<td>{data.id}</td>

<td>{data.name}</td>

<td>{data.username}</td>

<td>{data.email}</td>

</tr>

})

}

</tbody>

</table>

</div>

)

}

export default FetchApi;

step9:

-----

Test the application by using below request url.

ex:

http://localhost:3000

**React Redux**

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

**Redux is an open-source JavaScript library for managing and centralizing application state.**

**IT can be used with any frontend frameworks like ReactJS, AngularJS, VueJs and etc.**

**components of react redux**

**-------------------------**

**We have mainly three components for react redux.**

**1) Store :**

**-----------**

**Redux store is used to store entire state of our application.**

**2) Action :**

**-----------**

**It is only the way our application interact with redux store.**

**It carry some information from our application to redux store.**

**3) Reducer :**

**----------**

**Reducer read the payloads from the actions and then updates the store.**

**It is a pure function to return a new state from the initial state.**

react-redux

|

|---node\_modules

|

|---public

| |

|---favicon.ico

|---index.html

|---manifest.json

|-----src

| |

|---index.js

|---App.js

|

|---components

|

|---Counter.js

|

|---redux

|

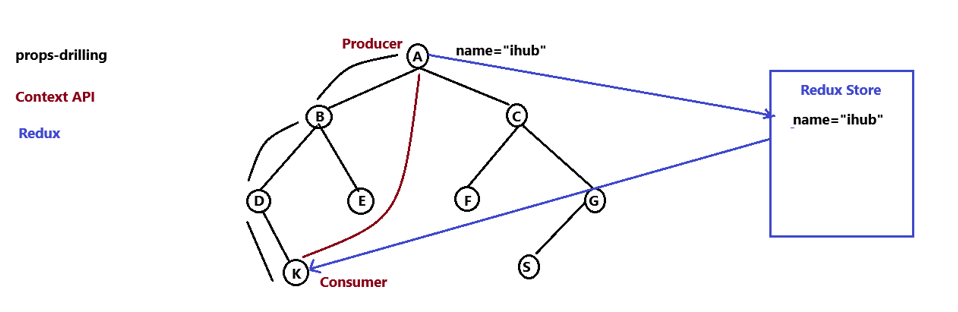
|---CounterAction.js

|---CounterReducer.js

|-----package.json

|-----README.md

Diagram:



step1:

------

Create a react application or project.

ex:

npx create-react-app my-redux-app

step2:

------

Switch to the project.

ex:

cd my-redux-app

step3:

-----

Install bootstrap , react, react-redux and redux library.

ex:

npm install bootstrap

npm install react react-dom react-redux redux

npm install @reduxjs/toolkit

Note:

----

import bootstrap inside index.js file.

step4:

-----

Run the react application.

ex:

npm start

step5:

------

Create Counter.js inside "components" folder.

Counter.js

-----------

import React from 'react'

export default function Couter() {

return (

<div className='container mt-5'>

<h2> Counter Application </h2>

<div class="container">

<button className='btn btn-primary'>Increment</button>

<b style={{fontSize:"30px"}} className="mx-3">{0}</b>

<button className='btn btn-warning'>Decrement</button>

</div>

</div>

)

}

step6:

-------

Create a CounterAction.js file inside "redux" folder.

CouterAction.js

---------------

export function Increment()

{

return{

type: "INCREMENT"

}

}

export function Decrement()

{

return {

type: "DECREMENT"

}

}

step7:

------

Create CounterReducer.js file inside "redux" folder.

CounterReducer.js

------------------

export function CounterReducer(state=0,action)

{

switch(action.type)

{

case "INCREMENT":

return state+1;

case "DECREMENT":

return state-1;

default:

return state;

}

}

step8:

------

Import useDispatch hook inside "Counter.js" to call actions.

Counter.js

---------

import React from 'react'

import {useDispatch} from 'react-redux'

import { Increment, Decrement } from '../redux/CouterAction'

export default function Couter() {

const dispatch=useDispatch();

return (

<div className='container mt-5'>

<h2> Counter Application </h2>

<div class="container">

<button className='btn btn-primary' onClick={()=>dispatch(Increment())}>Increment</button>

<b style={{fontSize:"30px"}} className="mx-3">{0}</b>

<button className='btn btn-warning' onClick={()=>dispatch(Decrement())}>Decrement</button>

</div>

</div>

)

}

step9:

------

Now to retrieve the data we need to declare the store in App.js file.

App.js

-------

import React from 'react'

import Couter from './components/Couter'

import { Provider } from 'react-redux'

import { configureStore} from '@reduxjs/toolkit'

import { CounterReducer } from './redux/CounterReducer'

const store= configureStore({

reducer:{

counter: CounterReducer

}

})

export default function App() {

return (

<Provider store={store}>

<Couter/>

</Provider>

)

}

step10:

------

To display the data we need to write below code in Counter.js file.

Counter.js

-----------

import React from 'react'

import {useDispatch, useSelector} from 'react-redux'

import { Increment, Decrement } from '../redux/CouterAction'

export default function Couter() {

const dispatch=useDispatch();

const counter=useSelector(state=> state.counter)

return (

<div className='container mt-5'>

<h2> Counter Application </h2>

<div class="container">

<button className='btn btn-primary' onClick={()=>dispatch(Increment())}>Increment</button>

<b style={{fontSize:"30px"}} className="mx-3">{counter}</b>

<button className='btn btn-warning' onClick={()=>dispatch(Decrement())}>Decrement</button>

</div>

</div>

)

}

======-----------=====================--------------------=====================-----------------------===

**Set interval** is method used to set time to perfrom some operation

 **useState**: Allows you to add state to functional components.

 **useEffect**: Performs side effects in functional components, like data fetching or subscriptions.

 **useContext**: Provides a way to consume values from a React context in a functional component.

 **useReducer**: An alternative to useState for managing complex state logic with reducers.

 **useMemo**: Memoizes a value to optimize performance by avoiding expensive calculations on re-renders.

 **useCallback**: Memoizes a function to prevent unnecessary re-creations during re-renders.

 **useRef**: Creates a mutable reference to an element or value that persists across re-renders.

 **useLayoutEffect**: Similar to useEffect, but runs synchronously after all DOM mutations, useful for measuring layouts.

 **useImperativeHandle**: Customizes the instance value exposed when using ref in parent components.

 **useDebugValue**: Displays a label in React DevTools for custom hooks to aid in debugging.

Imp questions in react

1. What is ReactJS?
2. Advantages of ReactJS?
3. What is JSX?
4. What is React Fragment?
5. What is React Component?
6. Function Component vs Class Component?
7. Write a program on Function Component and Class Component.
8. Real DOM vs Virtual DOM
9. Props vs State
10. Var vs Let vs Const
11. Phases of React Component?
12. Life Cycle Methods in React?
13. What is a Hook?
14. What is React Router?
15. What is SPA (Single Page Application)?
16. What is React Event Handling?
17. What is React Redux?
18. Render vs Re-render?
19. What is Reconciliation?
20. What is the map() Function?
21. What is the filter() Function?
22. What is the reduce() Function?
23. Controlled Component vs Uncontrolled Component (Forms)