ReactJS

========

It is a declarative, efficient, flexible javascript frontend library responsible to develop frontend applications and User interfaces i.e UI.

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

It is developed by Jordon Walke who was the software engineer at Facebook.

It was initially given by Facebook and later they have used in their own products like whatsapp and instagram.

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

The official website of reactjs is https://react.dev

The latest version of reactjs is v18.2.0.

The main objective of reactjs is to develop 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 is used to create reusable components.

3) It supports virtual DOM.

4) It supports one way data binding.

5) It supported by all major browsers.

6) Good documentation and community support.

Q) What is the difference between Angular and React ?

Angular	React
It is a product of Google.	It is a product of Facebook.
It was developed in Oct,2015.	It was developed in May,2013.
It is an open source javascript framework which is used to develop web and mobile applications.	It is an open source javascript frontend library responsible for view layer of an applications.
It uses traditional DOM.	It uses virtual DOM.
It supports two way data binding.	It supports one way data binding.

Angular is used to develop rich featured applications.	React is used to develop single page applications(SPA).
Typescript language is used.	JSX language is used.
Jasmine and Karma is used as a testing framework.	Jest and Enzyme is used as a testing framework.
It runs on 4200 port number.	It runs on 3000 port number.
Angular used by Google, Mc'Donalds, Nike and etc.	React us by Facebook, whatsapp, instagram, airbnb and etc.

Pre-requision to learn ReactJS

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

1) Knowledge on HTML,CSS,JavaScript and Bootstrap.

2) Basics on JSX

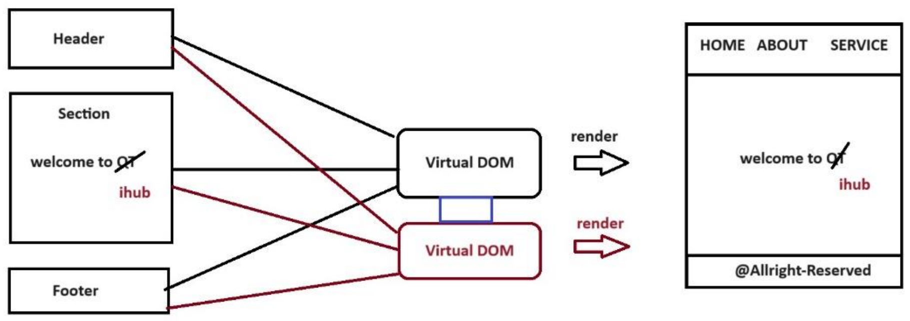
3) Idea on ES6

4) npm commands

How ReactJS works internally

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

Diagram: react1.1



ReactJS internally uses virtual DOM.

Virtual DOM is also a Tree Node Structure.

Virtual DOM will find a effective way to make the changes in traditional/real DOM.Hence react applications will execute fastly.

Assignment program

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

Q) Write a java program to display given output?

Input:

```
IJK
```

Output:

```
IJ
IK
```

```
	JI
	JK
	KI
	KJ
```

ex:

```
class Test
{
	public static void main(String[] args)
	{
		String str="IJK";

		int n=str.length();

		for(int i=0;i<n;i++)
		{
			for(int j=0;j<n;j++)
			{
				if(i!=j)
				{
					System.out.println(str.charAt(i)+""+str.charAt(j));
				}
			}
		}
	}
}
```

JSX

=====

JSX stands for JavaScript XML.

JSX allows us to write HTML code in Javascript.

JSX element contains tags, attributes and childrens.

JSX is a not a neccessity to create react application instead we can use Babel.

JSX makes our program simpler and elegant.

JSX ultimately transpile to pure javascript which is understand by a browser window.

JSX elements

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

JSX elements allows us to write HTML code without using createElement() method or appendChild() method.

ex:1

```
        JSX
        ----
                <h1> Heading Tag </h1>
        Babel
        -----
                React.createElement('h1',null,'Heading Tag');
```

Here

'h1' is a tag name

'null' is a optional attribute

'Heading Tag' is a children

ex:2

JSX

```
<div>
        <h1>Heading Tag</h1>
</div>
```

Babel

```
React.createElement('div',null,
        React.createElement('h1',null,'Heading Tag'));
```

ex:

JSX

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

Babel

```
                    React.createElement('h1',{id:'myId'},'Heading Tag');
```

ex:

```
          JSX
          ---
                    <h1 className="myClass"> Heading Tag </h1>
          Babel
          -----
                    React.createElement('h1',{class:'myClass'},'Heading Tag');
```

ex:

```
          JSX
          ---
                    <h1 id="myId" className="myClass"> Heading Tag </h1>

          Babel
          -----
          React.createElement('h1',{id:'myId',class:'myClass'},'Heading Tag');
```

JSX Expression

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

JSX expression is used to represent expression in curly brace i.e. {}.

JSX expression can be a variables, constants and any valid javascript expressions.

ex:1

```
        var a=10;

        <h1>{a}</h1>
```

ex:2

```
        <h1>{10 + 20 }</h1>
```

ex:3

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

npm

======

npm stands for Node Package Manager.

It is a integrated tool for nodejs.

npm is used to install node dependencies/packages/libraries.

We can install node dependencies as follow.

ex:

```
npm install -g  dependency_name/library/package_name
```

All the dependencies will be installed in "node_modules" folder.

Steps to work with npm

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

step1:

Download and Install nodejs software.

ex:

https://nodejs.org/en

step2:

Copy nodejs directory.

ex:

```
C:\Program Files\nodejs
```

step3:

Paste nodejs directory in environmental variables.

ex:

right click to my pc --> properties --> advanced system settings

---> environmental variables

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

variable name : path

variable value :

C:\Program Files\nodejs;C:\Users\Dell\AppData\Roaming\npm;

step4:

Check the environmental setup done perfectly or not.

ex:

```
cmd> node  -v
cmd> npm -v
```

step5:

Install npm by using below command.

ex:

```
cmd> npm install -g npm
or
cmd> npm install -g create-react-app
```

Steps to develop First React application

==

step1:

Make sure Nodejs setup is done perfectly.

step2:

Download and Install VSC (Visual Code Editor) editor.

ex:

https://code.visualstudio.com/

step3:

Create a "ReactProjects" folder inside "E" drive.

step4:

Open the command prompt from "Reactprojects" folder.

step5:

Open VSC editor from "Reactprojects" folder.

ex:

```
Reactprojects> code  .
```

step6:

Create a react application/project i.e myapp1.

ex:

```
Reactprojects>npx create-react-app  myapp1
```

step7:

Switch to the react project.

ex:

```
Reactprojects> cd  myapp1
```

step8:

Run the react project by using below command.

ex:

```
Reactprojects/myapp1> npm start
```

step9:

Test the application by using below request url.

ex:

http://localhost:3000

c

Note:

React application runs in a light weight development server with default

3000 port no.

Interview Questions

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

Q) How to create a react project or application?

```
npx  create-react-app   myapp1
```

Q) How to switch to the project.

```
cd   myapp1
```

Q) How to run react application or project.

```
npm start
```

Q) How to test the react application or project.

```
http://localhost:3000
```

React Project Structure and work flow

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

```
myapp1
|
|---node_modules
|
|---public
        |
        |---favicon.ico
        |---index.html
```

```
          |---manifest.json
|
|-----src
          |
          |---index.js
          |---index.css
          |
          |---App.js
          |---App.css
          |
          |---App.test.js
|
|------package.json
|------README.md
```

A "node_modules" contains all dependencies and libraries installed.

A "favicon.ico" is a favrouite icon of a react application.

A "index.html" is a main tempate of react application.

A "manifest.json " file contains metadata which is used when we install application on client mobile or computer.

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

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

A "App.js" is a parent component.

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

A "App.test.js" file is releted to unit testing.

A "package.json" file contains dependencies along with versions.

Note:

index.html	- main template
index.js	- entry point
App.js	- parent component
package.json	- dependencies with versions

Work flow

=========

```
          code load to        render to          output
App.js  --------------> index.js  ----------> index.html -------------> Browser
```

Steps to develop second application in react

===

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 myapp2 project.

ex:

```
ReactProjects> cd  myapp2
```

step4:

Run the react application.

ex:

```
ReactProjects/myapp2> npm start
```

step5:

Test the application by using below request url.

ex:

http://localhost:3000

step6:

Write below code in App.js file.

Approach1

App.js

```
function App()
{
  return(
    <h1>I Love ReactJS </h1>
  )
}
export default App;
```

Approach2

App.js

```
var App=function(){

  return(
    <h1>I Love ReactJS Programming</h1>
  )
}
export default App;
```

Approach3

App.js

```
var App=()=>{

  return(
    <h1>I Love ReactJS Programming and Development</h1>
  )
}
export default App;
```

React is mainly used to develop reusable components.

ex:

App.js

```
function App() {
  return (
    <h1>
      React Example for Reusability
    </h1>
  )
}
export default App
```

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 />
    <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

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

Fragment is used to group of list of childrens without adding

extra nodes of the DOM.

In general, We can return only one element at a time but we can't return more then one element directly.

To return more then one element we need to use React Fragment.

syntax

```
<React.Fragment>

        -

        -

</React.Fragment
```

or

```
<>

        -

        -

</>
```

Examples

App.js

```
function App
{
   return (
      //return react element
      return  <h1>IHUB Talent</h1>
            <h2>React Tutorial For Freshers</h2>
   );
}
//export React component
export default App
```

```
o/p: Filed to compile
```

To overcome above problem we can use <div> tag and inside that

<div> tag we can declare any child tags.

ex:

App.js

```
function App
{
   return (
      //return react element
      return
                  <div>
                           <h1>IHUB Talent</h1>
                  <h2>React Tutorial For Freshers</h2>
                  </div>
   );
}
//export React component
export default App
```

Note:

In above program "<div>" tag is a unused tag.

To remove unused/unnecessary tags we can use React Fragment.

approach1

App.js

```
import React from "react";

function App()
{
   return (
         <React.Fragment>
         <h1>IHUB React Tutorial</h1>
         <h1>React Classes for Freshers</h1>
         </React.Fragment>
   );

}
export default App;
```

approach2

App.js

```
import React from "react";
import {Fragment} from 'react';
function App()
{
   return (
         <Fragment>
         <h1>IHUB React Tutorial</h1>
```

```
            <h1>React Classes for Freshers</h1>
            </Fragment>
    );

}
export default App;
```

approach3

App.js

```
import React from "react";

function App()
{
    return (
            <>
            <h1>IHUB React Tutorial</h1>
            <h1>React Classes for Freshers</h1>
            </>
    );

}
export default App;
```

React Components

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

A component is a building block of react application.

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

ex:

    <Header> , <Footer>, <Section>, <Table>, <Form> and etc.

React components are like javascript functions because they accept arbitary inputs like props and return react element describing what should appear on the screen.

React component name always starts with uppercase letter.

There are two ways to declare react components.

1) Function Component / Functional Component

2) Class Component

1) Function Component

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

Function component is a javascript function which takes props as a argument along with inputs.

Functoin component is also known stateless component because it does not hold state.

syntax:1

```
function App()
{
        return
        (
                <h1> Named Function </h1>
        )
}
export default App;
```

syntax:2

```
var App=function()
{
        return
        (
                <h1> Anonymous Function </h1>
        )
}
export default App;
```

syntax:3

```
var App=()=>
{
        return
        (
                <h1> Arrow Function </h1>
        )
}
export default App;
```

Project structure

```
-----------------
myapp3
|
|---node_modules
|
|-----public
        |
        |---manifest.json
        |---index.html
        |---favicon.ico
|
|------src
        |
        |---index.js
        |
        |---App.js

|------package.json
|------README.md
```

step1:

------

create a react application i.e myapp3.

ex:

```
Reactprojects> npx  create-react-app myapp3
```

step2:

Open VSC editor from Reactprojects folder.

ex:

```
Reactprojects> code  .
```

step3:

Jump/Switch to myapp3 project.

ex:

```
Reactprojects> cd   myapp3
```

step4:

Run the react application.

ex:

```
Reactprojects/myapp3> npm start
```

step5:

Test the react application by using below request url.

ex:

http://localhost:3000

step6:

Declare below code inside App.js file.

App.js

```
var App=()=>{
return (
        <h1>Arrow Function component</h1>
)
}
export default App
```

Function component with props

In order to use props in a component We need to perform following changes in react "myapp3" project.

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
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();



App.js
------
var App=(props)=>{
  return (
    <>
    <h1>Name : {props.name}</h1>
    <h1>RollNo : {props.rollno}</h1>
    </>
  )
}
```

export default App

2) Class component

A class Component requires to extends from React Component.

The class must implements a render() method function which returns A react

Element to be render.This is Similar to return value of a functional

component.

In a class based component props are accessible via this.props.

The class component is also known as a stateful component because they can hold or manage local state.

Project structure

```
myapp4
|
|---node_modules
|
|-----public
        |
        |---manifest.json
        |---index.html
```

```
            |---favicon.ico
|
|------src
            |
            |---index.js
            |
            |---App.js

|------package.json
|------README.md
```

step1:

------

create a react application i.e myapp4.

ex:

```
Reactprojects> npx  create-react-app myapp4
```

step2:

Open VSC editor from Reactprojects folder.

ex:

```
Reactprojects> code  .
```

step3:

Jump/Switch to myapp4 project.

ex:

```
Reactprojects> cd   myapp4
```

step4:

Run the react application.

ex:

```
Reactprojects/myapp4> npm start
```

step5:

Test the react application by using below request url.

ex:

```
http://localhost:3000
```

step6:

Declare below code inside App.js file.

ex:

App.js

```
import {Component} from 'react';
class App extends Component
{
        render()
        {
        return(
        <h1>Class Component</h1>
        )
        }
}
export default App
```

Class component with props

----

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App name="Jose" rollno="501"/>
  </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();
```

App.js

```
import {Component} from 'react';
class App extends Component
{
   render()
   {
      return(
        <>
          <h1>Name :{this.props.name}</h1>
          <h1>RollNo :{this.props.rollno}</h1>
        </>
      )
   }
}
export default App
```

Composing Components in React

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

A component can refer to other components in their output is called composing component.

Let us use some component abstraction for any level of details.

Project structure

```
-----------------
myapp4
|
|-----node_modules
|
|-----public
|          |
           |---index.html (main template)
           |---favicon.ico (favicon)
           |---manifest.json (metadata)
|
|-----src
|          |
           |---index.js (entry point)
           |
           |
           |---App.js  (parent component)
           |
           |---Student.js (custom component)

|
|-----package.json
|-----README.md
```

step1:

Create a React Application.

ex:

```
ReactProjects>npx create-react-app  myapp4
```

step2:

Start Visual Studio Code (VSC) Editor.

ex:

```
ReactProjects> code   .
```

step3:

Delete all the files from "src" folder.

step4:

Create "index.js" file inside "src" folder.

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
```

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

root.render(
   <React.StrictMode>
      <App/>
   </React.StrictMode>
)
```

step5:

Create App.js file inside "src" folder.

```
App.js
--------
import Student from './Student';

function App()
{
   return (
      <Student/>
   )
}
export default App;
```

step6:

Create Student.js file inside "src" folder.

Student.js

```
function Student()
{
   return (
      <h1>Student Component</h1>
   )
}
export default Student;
```

step7:

Move to myapp4.

ex:

```
ReactProjects> cd  myapp4
```

step8:

Run the react application.

ex:

```
ReactProjects/myapp4> npm start
```

step9:

Check the output by using below url.

ex:

http://localhost:3000

composing components using props

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

index.js

```
import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App";

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

root.render(
   <React.StrictMode>
       <App course="React"/>
   </React.StrictMode>
)
```

App.js

```
import Student from './Student';

function App(props)
```

```
{
   return (
      <Student crs={props.course}/>
   )
}
export default App;
```

Student.js

```
function Student(props)
{
   return (
       <h1>My Course Name : {props.crs}</h1>
   )
}
export default Student;
```

React CSS

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

CSS in React is used to style the React App or Component.

There are two ways available to add styling to your React App or Component with CSS.

1) Inline Styling

2) CSS Stylesheet

1)Inline CSS

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

Inline CSS represent by "style" attribute in React application.

The inline styles are specified with a JavaScript object in camelCase version of the style name.

ex:

App.js

```
import Student from "./Student";

function App()
{
  return <>
        <h1 style={{color:"green"}}>React Inline CSS</h1>
        <h1 style={{backgroundColor:"yellow"}}>React Inline CSS</h1>
         </>
}
export default App;
```

The inline styling also allows us to create an object with styling information and
refer it in the style attribute.

App.js

```
import Student from "./Student";


function App()
{
  const mystyle = {
    color: "white",
    backgroundColor: "DodgerBlue",
    padding: "10px",
    fontFamily: "Arial"
  };
  return <>
         <h1 style={mystyle}>React Inline CSS</h1>
         <h1 style={{backgroundColor:"yellow"}}>React Inline CSS</h1>
          </>
}
export default App;
```

2) CSS Stylesheet

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

We can write styling in a separate file for your React application, and save the file with a .css extension.

Later we can import .css file in our required application.

ex:1

App.js

```
import Student from "./Student";
import './App.css';
function App()
{

  return <>
        <h1>React CSS styles</h1>
        <h1>React CSS styles</h1>
         </>
}
export default App;
```

App.css

```
body{
  background-color: yellow;
}
h1
{
  color:blue;
}
```

ex:2

App.js

```
import Student from "./Student";
import './App.css';
function App()
{

  return <>
        <h1 id="myId">React CSS styles</h1>
        <h1 className="myClass">React CSS styles</h1>
         </>
}
export default App;
```

App.css

```
body{
  background-color: yellow;
}
```

```
#myId
{
  color:blue;
}
.myClass
{
  color:red;
}
```

State
=======

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

we can create a state only in class component but not in functional component.

It is possible to update the state or modify the state , where as props
only for read only.

There are two ways to initialize the state in React component.

1)Directly inside class

2)Inside the Constructor

1)Directly inside class
=========================

class Student extends Component

```
{
	//define state
	state={
		name: "Anna Julie",
		prop1: this.props.prop1
	}
	render()
	{
		-
	}
}
```

Note:

The "state" property is refered as state.

"this" is a class instance property

example

Project structure

```
myapp6
|
|------node-modules
|
|------public
|	|
|          |------favicon.ico
|          |------index.html
|          |------manifest.json
|
|------src
           |
           |------index.js
           |------App.js
|
|------package.json
|------README.md
```

step1:

Develop React Application.

ex:

```
E:/ReactProjects>npx create-react-app  myapp6
```

step2:

Open VSC editor from Reactprojects.

ex:

```
E:/Reactprojects>code .
```

step3:

Install "ES7 React " Plugin/Extension from Visual Studio Code for shortcuts to create React Applications.

ex:

imr +tab

imrc + tab

imrd + tab

imp + tab

rcc - class component

rfce - named function component

rafce - anonymous function component

conlg+ tab

step4:

Add below code inside "App.js" file.

App.js

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

export default class App extends Component {
            state={
                  name:"Alan"
            }
  render() {
      return (
        <h1>Hello {this.state.name}</h1>
      )
```

```
 }
}
```

step5:

-----

    move to myapp5

    ex:

```
E:/BUI-2pm/ReactProjects> cd myapp6
```

step6:

    Run the application.

    ex:

```
DE:/BUI-2pm/ReactProjects/myapp6>npm start
```

step7:

    Test the React Application.

    ex:

```
http://localhost:3000
```

ex:2

App.js

```
-----------
import React, { Component } from 'react'

export default class App extends Component {
                        state={
                                    name:"Alan",
                                    roll:this.props.rollno
                        }
  render() {
            return (
              <div>
                        <h1>Name: {this.state.name}</h1>
                        <h1>RollNo: {this.state.roll}</h1>
              </div>
            )
  }
}
```

index.js

```
----------

import App from './App';
import ReactDOM from 'react-dom/client';
import React from 'react';

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

```
root.render(
    <React.StrictMode>
        <App rollno={501} />
    </React.StrictMode>
)
```

Note:

Here props property we are storing into a state.

2)Inside the Constructor

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

```
class App extends Component
{
        //constructor
        //props is optional
        constructor(props)
        {
                //it is required to call the parent class constructor
                super(props);
                //state
                this.state={
                        name:"alan",
                        prop1: this.props.prop1
                }
        }
        render()
        {
                -
        }
}
```

When the component class is created,The constructor is the first called
so it is right place to add state.

The class instance has already been created in memory .So we can use
"this" to set properties on it.

When we write a constructor ,make sure to call parent class constructor
by using super(props) keyword.

When we call super with props ,React will make props avaiable accross/access
the component through this.props.

Project structure

```
myapp7
|
|------node-modules
|
|------public
|          |
|          |------favicon.ico
|          |------index.html
|          |------manifest.json
|
|------src
           |
           |------index.js
           |------App.js
|
|------package.json
|------README.md
```

step1:

Develop React Application.

ex:

```
E:/ReactProjects>npx create-react-app  myapp7
```

step2:

Open VSC code editor.

ex:

```
ReactProjects> code .
```

step3:

Write below code inside "App.js" file in "src " folder (rcc).

Student.js

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

export default class App extends Component {

        constructor()
        {
                super();

                this.state={
                        name: "Alan",
                        roll: 101
                }
        }
  render() {
        return (
          <div>
                <h1>Name: {this.state.name}</h1>
                <h1>RollNo: {this.state.roll}</h1>
```

```
            </div>
          )
  }
}
```

step4:

-----

move to myapp7

ex:

```
E:/BUI-2pm/ReactProjects> cd myapp7
```

step5:

Run the application.

ex:

```
DE:/BUI-2pm/ReactProjects/myapp7>npm start
```

step6:

Test the React Application.

ex:

```
http://localhost:3000
```

ex:2

App.js

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

export default class App extends Component {

        constructor(props)
        {
                super(props);

                this.state={
                        name: "Alan",
                        roll: this.props.rollno
                }
        }
  render() {
        return (
          <div>
                <h1>Name: {this.state.name}</h1>
                <h1>RollNo: {this.state.roll}</h1>
          </div>
        )
  }
}
```

index.js

```
----------
import App from './App';
import ReactDOM from 'react-dom/client';
import React from 'react';

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <React.StrictMode>
        <App rollno={501} />
    </React.StrictMode>



)
```

Event Handling in React

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

Event

Action to which a javascript can respond is called event.

ex:

clicking on button

hovering of an element

and etc.

Handling events on react Elements are same like handling events on DOM elements.

ex:

Javascript

```
        <button  onclick="f1()">clickMe</button
```

React

```
        <button onClick={handleClick}>clickMe</button>  --> Function component

        <button onClick={this.handleClick}>clickMe</button>  --> Class component
```

Eventing Handling using Function component

==

Project structure

```
myapp8
|
|----node_modules
```

```
|
|----public
          |
          |---index.html
          |---favicon.ico
          |---manifest.json

|
|-----src
|         |
          |---index.js
          |---index.css

          |
          |---App.js
          |---App.css
          |---App.test.js
|
|----package.json
|
```

step1:

-----

create a react project/application.

ex:

```
ReactProjects>npx create-react-app  myapp8
```

step2:

Starts VSC code editor.

ex:

```
ReactProjects> code .
```

step3:

Move to the project.

ex:

```
ReactProjects> cd myapp8
```

step4:

Run the react application/project.

ex:

```
ReactProjects/myapp8> npm start
```

ex:1

App.js

```
function App()
{
  function handleClick()
  {
    console.log("Button is clicked");
  }



 return (
    <button onClick={handleClick}>clickMe</button>
 )
}
export default App;
```

ex:2

App.js

```
function App()
{

  const handleClick=()=>
  {
    console.log("Button is clicked");
  }
```

```
 return (
    <button onClick={handleClick}>clickMe</button>
 )
}
export default App;
```

ex:3

```
import React from 'react'

function App() {

  function handleClick(e)
  {
     e.preventDefault();
     console.log("You have clicked");
  }

  return (
   <div>
      <a href="http://www.google.com" onClick={handleClick}> click </a>
   </div>
  )
}

export default App
```

Eventing Handling using class component

==

Project structure

```
myapp9
|
|----node_modules
|
|----public
            |
            |---index.html
            |---favicon.ico
            |---manifest.json

|
|-----src
|           |
            |---index.js
            |---index.css


            |
            |---App.js
            |---App.css
            |---App.test.js
```

```
|
|----package.json
|
```

step1:

-----

create a react project/application.

ex:

```
ReactProjects> create-react-app  myapp9
```

step2:

Starts VSC code editor.

ex:

```
ReactProjects> code .
```

step3:

Move to the project.

ex:

```
ReactProjects> cd myapp9
```

step4:

Run the react application/project.

ex:

```
ReactProjects/myapp9> npm start
```

ex:1

App.js

```
import {Component} from "react";
export default class App extends Component
{


   handleClick=()=>
   {
      console.log("Button is clicked",this);
   }

   render()
   {
      return(
        <button  onClick={this.handleClick}>clickMe</button>
      )
   }
}
```

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 />
  </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();
```

update state

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

Using setState() method is used to update states.

ex:

```
this.state={
        name:"Alan"
}

this.setState({name:"Kelvin"});
```

ex:1

App.js

```
import {Component} from "react";
export default class App extends Component
{
   state={
      name : "Nancy",
      rollno: 101
   }

   handleClick=()=>
   {
     this.setState({name:"Lisa",rollno:501});
   }
  render()
  {
     return(
      <>
```

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

            <button  onClick={this.handleClick}>Change state</button>
          </>
        )
    }
}
```

Interview Questions

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

Q)Difference between function component vs class component?

function component	class component
It is also known as stateless component.	It is a statefull component.
In a function component we will use return keyword.	In a class component we will use render() method.
It supports hooks.	It does not support hooks.
Constructor is not used.	Constructor is used.

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.

Q)Difference between props and state ?

props	state
Props are read-only.	States are updatable.
Props are immutable.	State is mutable.
Props allow us to pass data from one component to other components as an argument.	State holds information about the components.
Props can be accessed by the child component.	State cannot be accessed by child components because it is private.

Stateless component can have Props.	Statefull components can have state.

Phases of components in ReactJS

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

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.

2)Updating

Updating is a process of changing state or props of a component and update changes to nodes already existing 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 tree.

In general it will clear the reserved memory.

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()

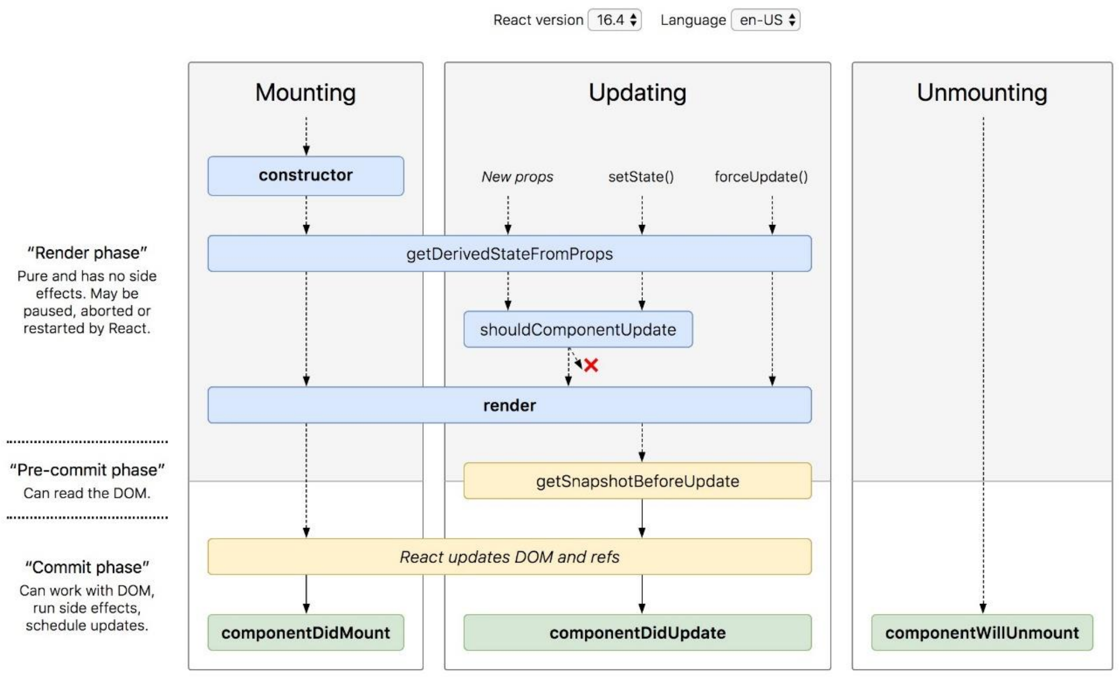
2) shouldComponentUpdate()

3) render()

4) getSnapshotBeforeUpdate()

4) ComponentDidUpdate()

Diagram: react6.1



In react , all life cycle methods we can declare inside class component.

App.js

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

export default class App extends Component {

  constructor()
  {
    console.log('constructor');
     super();

      this.state={
        name:"Alan"
      }
  }

  static getDerivedStateFromProps(props,state)
  {
    console.log('getDerivedStateFromProps')
  }

  render() {
    console.log('render');
    return (
      <>
        <h1>Name : {this.state.name}</h1>
      </>
    )
  }

  componentDidMount()
  {
    console.log('componentDidMount')
  }
}
```

Hooks

==========

Hooks allow us to "hook" into React features such as state and lifecycle methods.

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

Hooks allow us to use React without classes.It means you can use state and other React features without writing a class.

React provides a few built-In hooks like useState,useEffect and etc.

Hooks are new addition in React 16.8.

When use Hooks

If you write a function component and relize you need to add some state to it.

Rules of Hooks

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

There are 3 rules for hooks:

1)Hooks can only be called inside React function components.

2)Hooks can only be called at the top level of a component.

3)Hooks cannot be conditional

Note: Hooks will not work in React class components.

Declaring State

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

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

We call it inside a function component to add some local state to it.

A useState() returns a pair - the current state value and a function that let us update it.

React will preserve this state between re-renders.

We can call this function from an event handler or somewhere else.

Project structure

```
--------------------
myapp10
|
|---node_modules
|
|---public
          |
          |--favicon.ico
          |--index.html
          |--manifest.json
|
|------src
          |
          |---App.js
          |
          |---index.js

|-----package.json
|-----README.md
```

step1:

----

create a react project.

ex:

```
Reactprojects> npx create-react-app myapp10
```

step2:

Open the VSC editor.

ex:

```
Reactprojects> code .
```

step3:

Move/Jump to myapp8 project.

ex:

```
Reactprojects> cd myapp10
```

step4:

Run the myapp8 project.

ex:

```
Reactprojects/myapp10> npm start
```

step5:

Test the application by using below request url.

ex:

http://localhost:3000

App.js

```
import { useState } from "react";
function App()
{
   const [name,setName]=useState("Alan");

   const handleClick=()=>
   {
        setName("Kelvin");
   }
   return (
      <div>
         <h1>Name : {name}</h1>
         <button onClick={handleClick}>clickMe</button>
      </div>
   )
}
export default App;
```

index.js

```
import Student from './Student';
import ReactDOM from 'react-dom/client';
import React from 'react';
import App from './App';

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <React.StrictMode>
         <App />
    </React.StrictMode>
)
```

Effect Hooks

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

The Effect Hook let us to perform side effects in function components.

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

useEffect()

===========

A useEffect is a hook for encapsulating code that has "side effects".
if we are familiar with React class life cycle methods.We can thing of useEffect Hooks as componentDidMount,compnoentDidUpdate and componentWillUnmount combined.

useEffect =componentDidMount+ componentDidUpdate +componentWillUnmount

ex:

```
import React,{useEffect} from "react";
useEffect(Function)
or
useEffect(Function ,Array)
```

The function passes to useEffect will run after the render is committed to the screen.

Second argument to useEffect that is the array of values that the effect depends on.(It is for condition purpose).

Note:

We can call useEffect as many times we required.

ex:

```
useEffect(()=>
{
        console.log("Hello useeffect");
});
```

ex:

```
useEffect(()=>
{
        console.log("Hello useEffect");
},[count]);
```

What does useEffect do?

By using this Hook,we can tell react that your component needs to do something after render.

React remember the function we passed and call it later after performing the DOM updates.

In this effect, we set the document title,we could also perform data fetching or call some other imperative API.

Note:

----

useEffect runs after the first render and after every update.

Project structure

```
myapp11
|
|---node_modules
|
|---public
        |
        |--favicon.ico
```

```
            |--index.html

            |--manifest.json

|

|------src

            |

            |---App.js

            |

            |---index.js


|-----package.json

|-----README.md
```

step1:

----

create a react project.

ex:

```
Reactprojects> npx create-react-app myapp11
```

step2:

Open the VSC editor.

ex:

```
Reactprojects> code .
```

step3:

Move/Jump to myapp9 project.

ex:

```
Reactprojects> cd myapp11
```

step4:

Run the myapp9 project.

ex:

```
Reactprojects/myapp11> npm start
```

step5:

Test the application by using below request url.

ex:

```
http://localhost:3000
```

App.js

```
import { useState, useEffect } from "react";
function App()
{
   const [count,setCount]=useState(0);

   const handleClick=()=>
   {
        setCount(count+1);
    }

    useEffect(() => {
      // Update the document title using the browser API
      document.title = `you have click for ${count} times`;
     });
    return (
      <div>
          <h1>You clicked  {count} Times</h1>
          <button onClick={handleClick}>clickMe</button>
      </div>
    )
}
export default App;
```

index.js

```
import Student from './Student';
import ReactDOM from 'react-dom/client';
import React from 'react';
import App from './App';
const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
     <React.StrictMode>
          <App />
```

```
    </React.StrictMode>
)
```

React useContext Hook (Context API)

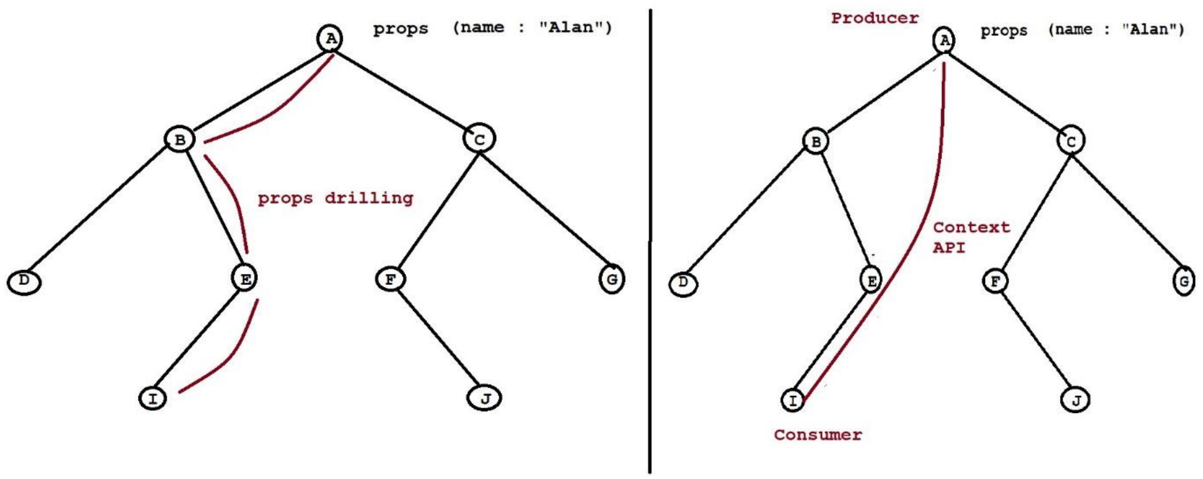
=====================================

Context provides a way to pass the data through the component tree without

passing props down manually at several level.

To do this without Context, we will need to pass the state(useState) as "props" through each nested component. This is called "props drilling".

Diagram: react7.1



Project structure

----

myapp12

|

```
|------node-modules
|
|------public
|          |
|          |------favicon.ico
|          |------index.html
|          |------manifest.json
|
|------src
           |
           |------index.js
           |------App.js
           |------Acomponent.js
           |------Bcomponent.js
           |------Ccomponent.js
|
|------package.json
|------README.md
```

Diagram: react7.2

step1:

----

create a react project.

ex:

```
Reactprojects> npx create-react-app myapp12
```

step2:

Open the VSC editor.

ex:

```
Reactprojects> code .
```

step3:

Move/Jump to myapp10 project.

ex:

```
Reactprojects> cd myapp12
```

step4:

Run the myapp10 project.

ex:

```
Reactprojects/myapp12> npm start
```

App.js

```
import React from 'react';
import Acomponent from "./Acomponent";
export const UseContext=React.createContext();
function App()
{
  return (
    <div>
      <UseContext.Provider value={'IHUB'}>
      <Acomponent/>
      </UseContext.Provider>


    </div>
  )
}
export default App;

Acomponent.js
---------------
import Bcomponent from "./Bcomponent";

function Acomponent()
{
  return (
    <Bcomponent/>
  )
}
export default  Acomponent;



Bcomponent.js
----------------
import Ccomponent from "./Ccomponent";
```

```
function Bcomponent()
{
   return (
      <Ccomponent/>
   )
}
export default  Bcomponent;
```

Ccomponent.js

```
import {UseContext} from "./App";

function Ccomponent()
{
   return (
   <div>
    <UseContext.Consumer>
        {
            user => {
                return <div>The value is : {user} </div>
            }
        }
    </UseContext.Consumer>
   </div>
   )
}
export default  Ccomponent;
```

index.js

```
import Student from './Student';
import ReactDOM from 'react-dom/client';
import React from 'react';
import App from './App';

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <React.StrictMode>
        <App />
    </React.StrictMode>
)
```

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

```
-------------------
myapp11
|
|----node_modules
|
|----public
|         |
          |----favicon.ico
          |----index.html
          |----manifest.json
|
|-----src
|         |
          |----index.js
          |----App.js
          |----CustomHook.js
|
|-----package.json
|-----README.md
```

step1:

Create a react project or application.

ex:

```
Reactprojects> npx create-react-app myapp11
```

step2:

Open VSC editor.

ex:

```
Reactprojects> code .
```

step3:

Move or Jump to myapp11 project.

ex:

```
Reactprojects> cd  myapp11
```

step4:

Run the react application.

ex:

```
Reactprojects/myapp11> npm start
```

step5:

Test the react application.

ex:

http://localhost:3000

step6:

Create "CustomHook.js" file inside "src" folder.

ex:1

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



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 />
  </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();
```

Images/Assets in ReactJS

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

We can set images/Asset in ReactJS using two ways.

1)Inside public Folder.

2)Inside src folder.

1)Inside public folder

If we put a file into a public folder,It will not be processed by webpack. Instead it will be copied into the build folder untouched.

To reference assets in the public folder, we need to use a special variable called PUBLIC_URL. Only files inside the public folder will be accessible by %PUBLIC_URL% prefix.

How to use image

----

1)

```
myapp13
|
|---public
          |
          |---rock.jpg
```

index.html

```
<img src="%PUBLIC_URL%rock.jpg" alt="mypic"/>
```

2)

```
myapp
|
|---public
            |
            |---images
                        |
                        |--rock.jpg
```

index.html

```
<img src="%PUBLIC_URL%/images/rock.jpg" alt="mypic"/>
```

If we want to use Image in Javascript file.

App.js

```
<img src={process.env.PUBLIC_URL +"/rock.jpg" } />
<img src={process.env.PUBLIC_URL +"/images/rock.jpg" } />
```

ex:1

```
index.html
--------
-
-
-
 <div id="root"></div>
<img src="%PUBLIC_URL%/team1.jpeg" alt="mypic"/>
-
-
-
```

Note:

Mostly of the time we are displaying images in Component only.

ex:

```
App.js
------
import React, { Component } from 'react'

export default class App extends Component {
        render() {
                return (
                        <div>
<img src={process.env.PUBLIC_URL+"team1.jpeg"} alt="mypic"></img>
                        </div>
                )
        }
}

index.js
----------
import React from 'react';
import ReactDOM from 'react-dom';
import App from "./App";

//render the component to index.html
ReactDOM.render(<App />,document.getElementById("root"));
```

2)Inside src folder

----

we can import a file right in a Javascript module.This tell webpack to include that file in the bundle.

How to use

----

1)

```
myapp
|
|---src
        |
        |---rock.jpg
App.js
-----
```

```
import pic from "./rock.jpg";
<img src={pic} alt="mypic" />
```

This ensures that when the project is built.Webpack wil correctly move the images into the build folder and provide us with correct paths.

ex:

App.js

```
import React, { Component } from 'react'
import pic from "./team1.jpeg";

export default class App extends Component {
        render() {
                return (
                        <div>
                                <img src={pic} alt="mypic"></img>
                        </div>
                )
        }
}
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from "./App";
```

```
//render the component to index.html
ReactDOM.render(<App />,document.getElementById("root"));
```

Interview Question

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

Q) Write a java program to display the string in a given format?

```
input:
        Hello55
        World5
Output:
        HelloWorld60
```

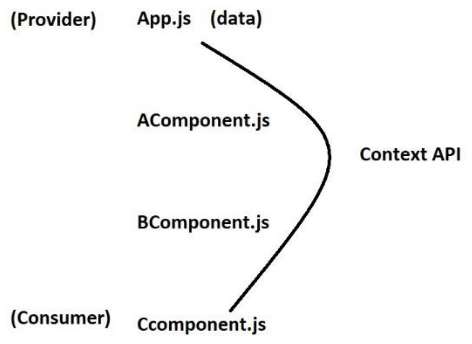
ex:

```
class Test
{
        public static void main(String[] args)
        {
                String str1="Hello55";
                String str2="World5";

                String word1=str1.replaceAll("[^A-Za-z]","");//Hello
                int num1=Integer.parseInt(str1.replaceAll("[^0-9]",""));//55
```

```
                    String word2=str2.replaceAll("[^A-Za-z]","");//World
                    int num2=Integer.parseInt(str2.replaceAll("[^0-9]",""));//5
                    String word=word1+word2;
                    int num=num1+num2;
                    System.out.println(word+num);
          }
}
```

Diagram 7.2



React Router

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

Routing is a process in which a user is directed to different pages based on their actions or requests.

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

```
myapp12
|
|------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 "myapp12" project in VSC.
        ex:
        projects>npx create-react-app myapp12
```

step2:

```
        Move to myapp12 project.
        ex:
        project> cd   myapp12
```

step3:

```
        install react router dom.
        ex:
        project/myapp12>npm install --save react-router-dom
```

step4:

Restart the application .

ex:

```
myapp14> 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>
);
```

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

```
myapp15
|
|----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 myapp15.

ex:

```
Reactprojects> npx create-react-app myapp15
```

step2:

Open the VSC code editor.

ex:

```
Reactprojects> code .
```

step3:

Move/Switch to myapp15 project.

ex:

```
Reactprojects> cd myapp15
```

step4:

Install Bootstrap package.

ex:

```
Reactprojects/myapp15> npm install bootstrap
```

step5:

Run the react application.

ex:

```
Reactprojecs/myapp15> 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.

A form can contain text fields, buttons, checkbox, radio button, etc.

Creating Form

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.

Controlled component

In the controlled component, the input form element is handled by the component rather than the DOM. Here, the mutable state is kept in the state property and will be updated only with setState() method.

Controlled components have functions that govern the data passing into them on every onChange event, rather than grabbing the data only once, e.g., when you click a submit button.This data is then

saved to state and updated with setState() method. This makes component have better control over the form elements and data.

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:

Switch/Move to myapp16 project.

ex:

```
Reactprojects> cd  myapp16
```

step4:

Install bootstrap package.

ex:

```
Reactprojects/myapp16> npm install bootstrap
```

step5:

Run the react application.

ex:

```
Reactprojects/myapp16> 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

Seguro interview Question

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

Write a java program to calculate costs based on user input. The program

should prompt users to enter the total weight of items(in kilograms) and the shipping destination (domestic or international). for demostic orders, the program should charge Rs.500 for weights upto 5 kg and Rs.100 per additional kg. for international orders, it should charge Rs.1000 for weights upto 5 kg , Rs.200 per additional kg , and a Rs.500 surcharge for weights exceeding 10 kg. print calculated shipping cost.

Input:

Total Weight of items : 12kg

shipping destination : domestic

<=5kg --> Rs. 500

>5kg --> Rs. 100 per kg

Output:

500 + 700 = 1200

Input:

Total Weight of items : 12kg

shipping destination : international

<=5kg --> Rs. 1000

>5kg ---> Rs. 200 per kg + extra 500

output:

1000 + 1400 + 500 = 2900

Program

```
import java.util.Scanner;
class Test
{
```

```
public static void main(String[] args)
{
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the total Weight of items :");
        int weight=sc.nextInt();

        System.out.println("Enter the shipping destination :");
        String destination=sc.next();

        int cost=shippingToCost(weight,destination);
        System.out.println("Total Cost Is :"+cost);
}
//static method
public static int shippingToCost(int weight,String destination)
{
        if(destination.equals("domestic"))
        {
                        if(weight<=5)
                                return 500;
                        else
                                return 500 + (weight-5) * 100;
        }
        else if(destination.equals("international"))
        {
                        if(weight<=5)
                                return 1000;
```

```
                    else if(weight<=10)
                        return 1000 + (weight-5) * 200;
                    else
                        return 1000 + (weight-5) * 200 + 500;
            }
            else
            {
                System.out.println("Invalid shipping address");
                return 0;
            }
        }
}
```

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

----

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:

Move/Switch to myapp17 project.

ex:

```
Reactprojects> cd  myapp17
```

step4:

Run the react application.

ex;

```
Reactprojects/myapp17> 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
```

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

----

myapp18

```
|
|----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 myapp18.

ex:

```
Reactprojects> npx  create-react-app  myapp18
```

step2:

Open the VSC code editor.

ex:

```
Reactprojects> code  .
```

step3:

Move/Switch to myapp17 project.

ex:

```
Reactprojects> cd  myapp18
```

step4:

Install axios in myapp18 project.

ex:

```
Reactprojects/myapp18> npm install axios
```

step5:

Run the react application.

ex;

```
Reactprojects/myapp18> 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