1. **Using any modern UI library, framework like React, Vue, etc. of your choice create a basic reusable component that displays Name, Email, Phone Number in a list view. The values for these fields (Name, Email, Phone Number) should be passed as a prop to the component.**

**Solution:**

**//App.js**

import React from 'react'

import Dibu from './Components/Dibu'

const App = () => {

  return (

    <Dibu

               listView =

                   {

                      [

                          {

                              Name:"person1",

                              Email:"person1@gmail.com",

                              phoneNumber : "124678"

                          },

                          {

                              Name:"person2",

                              Email:"person2@gmail.com",

                              phoneNumber : "12768"

                          },

                          {

                              Name:"person3",

                              Email:"person3@gmail.com",

                              phoneNumber : "264212"

                          }

                      ]

                   }

           />

  )

}

export default App

//Dibu.js

import React from "react";

const Dibu = (props)=>{

    return(

        <>

            {

                    props.listView.map(item=>{

                    return(

                        <>

                            <p> Name : {item.Name}</p>

                            <p> Email :  {item.Email}</p>

                            <p> PhoneNumber: {item.phoneNumber}</p>

                            <br/>

                        </>

                    )

                })

            }

        </>

    )

}

export default Dibu;

**2.Refer to the figure below: Create a parent component showing number 10 in the display. Create a child component maybe a button, which is inside this parent component and when user clicks on the button the number will be increased by 5 so new number 15 will be displayed in the parent component. You can extend this functionality on answer of question #1. Please use any library, framework you wish to solve this problem.**

**Solution:**

//App.js

import React, {useState} from "react";

import "./App.css"

const App=()=>{

  const[number, setNumber]=useState(10);

  return<div className="parent">

    <div className="content">

    <h1>The number is:{number}</h1>

    <button onClick={()=>setNumber(number+5)} className="btn">Add 5</button>

    </div>

    </div>

}

export default App;

//App.css

.parent{

  position: absolute;

}

.content{

  position: relative;

  left: 600px;

  padding: 10px 10px;

  margin-top: 10px;

  border: 1px solid black;

  text-align: center;     ;

}

.btn{

  margin-bottom: 20px;

}

**6. What is the output when I execute both the functions? Are they same/different? If yes/no then why?**

**function func1() {**

**return {**

**out: "hello" };**

**}**

**function func2() {**

**return**

**{**

**out: "hello" };**

**}**

**Solution:**

Func1 will return out: “hello” but the function2 will return “undefined” because of the automatic semicolon insertion. In above example, function func2 is translated into

function func2(){

return;

{

out:”hello”

}

The semicolon then terminates the return statement and therefore returns undefined.

**7. Assume that there are three buttons in a webpage with ids: button0, button1, button2. What will be the output of the following function for each button if the buttons are clicked.**

function hookupevents() {

for (var i = 0; i < 3; i++) {

document.getElementById("button" + i)

.addEventListener("click", function() {

alert(i);

});

}

}

***Solution:***

The browser displays a modal dialog with a message :

“0”, if button0 is pressed

“1”, if button1 is pressed

and “2”, if button2 is pressed, along with a OK button.

**8. Consider the following dataset representing sales of Sanitizer produced by a company in the last 7 days. Show this dataset with the help of any visualization library in JS of your choice.**

**Date Sales Count**

**2021-07-11 53**

**2021-07-12 42**

**2021-07-13 88**

**2021-07-14 14**

**2021-07-15 77**

**2021-07-16 68**

**2021-07-17 43**

**Solution:**

    import React from "react";

    import { PieChart, Pie, Legend, Tooltip } from "recharts";

    const data01 = [

    { name: "2021-07-11 ", value: 53 },

    { name: "2021-07-12", value: 42 },

    { name: "2021-07-13", value: 88 },

    { name: "2021-07-14", value: 14 },

    { name: "2021-07-15", value: 77 },

    { name: "2021-07-16", value: 68},

    { name: "2021-07-17", value: 43}

    ];

    export default function App() {

    return (

        <><h1 style={{marginLeft:"45%"}}>Sales</h1>

        <PieChart style={{marginLeft:"35%"}}width={2000} height={600}>

        <Pie

            dataKey="value"

            isAnimationActive={false}

            data={data01}

            cx={200}

            cy={200}

            outerRadius={80}

            fill="#8884d8"

            label/>

        <Tooltip />

        </PieChart></>

    );}

**9.What is the difference between Server Side Rendering (SSR) and Static Site Generation (SSG)? When would you prefer SSG over SSR?**

**Solution:**

In server side rendering, the user agent requests a page. Then the sever generates the page’s HTML output and sends it back. The browser renders the HTML page afterwards.

Meanwhile, in static site generation, there’s no need of server.

SSG is preferred over SSR if whenever we lack highly dynamic data that needs to be unique for every user.

**10. Create a simple webpage that shows the data fetched from an API response. You can use this dummy API endpoint (https://jsonplaceholder.typicode.com/users) to fetch the list of users and only show the name and email in a list view in your webpage. Please feel free to use any HTTP client of your choice. Some popular ones are axios, fetch, etc**.

Solution:

import React from 'react';

import axios from 'axios';

export default class UserList extends React.Component {

  state = {

    users: []

  }

  componentDidMount() {

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

      .then(res => {

        const users = res.data;

        this.setState({ users });

      })

  }

  render() {

    return (

      <ul>

        {

          this.state.users

            .map(user =>

              <>

              <li key={user.id}>{user.name}</li>

              <li key={user.id}>{user.email}</li>

              </>

            )

        }

      </ul>

    )

  }

}