       MODULE: 9 ReactJs Intro

**Que : 1  What is React Js?**

**Ans.**

         ReactJS is an open-source JavaScript library used to create user interfaces in a declarative and efficient way.  React is used to create modular user interfaces and promotes the development of reusable UI components that display dynamic data.

**Que : 2  What is NPM in React Js?**

**Ans.**

       NPM stands for Node Package Manager. NPM is the default package manager for Node.js, and it is used to manage and distribute packages or libraries of reusable code. React.js itself is a JavaScript library for building user interfaces, and it often relies on various external packages or modules to enhance its functionality or simplify common tasks.

        When working with React.js, you typically use NPM to install and manage third-party libraries or packages that are relevant to your project. These packages can include things like state management libraries (e.g., Redux), routing libraries (e.g., React Router), utility functions, and more.

**Que : 3  What is the Role of Node Js in React Js?**

**Ans.**

         Node is the most popular platform for hosting and running web servers for React apps. After all, using NPM (Node Package Manager), you can install any package with the NPM command-line interface CLI. Moreover, Node utilizes Webpack and other modules to turn a React application into an easily compilable single file.

**Que : 4  What is CLI command In React Js?**

**Ans.**

       React has its own command-line interface (CLI) commands. However, these CLI commands are currently only used to create a passable version of a react application using the command line. This will contain a default template as its design, so all the react applications created this way will have great consistency as they all have the same structure.

**Que : 5  What are Components in React Js?**

**Ans.**

       A Component is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier. You can see a UI broken down into multiple individual pieces called components and work on them independently and merge them all in a parent component which will be your final UI.

**Que : 6  What is Header and Content Components in React Js?**

**Ans.**

        In a React.js application, the terms "Header" and "Content" typically refer to components that represent specific sections of the user interface. These components are often used to structure the layout of a page and organize the presentation of information. Let's explore the concepts of Header and Content components in React:

**Header Component:**

            A Header component in a React application typically represents the top section of a page and contains elements such as the application title, navigation links, logos, and any other content that should appear at the top of the UI. The Header component is responsible for rendering the user interface elements related to the application's header.

**Content Component:**

             A Content component, on the other hand, represents the main content area of a page. It is responsible for rendering the primary content that the user interacts with or views. This component may include various sub-components, data fetching logic, or other elements that make up the main body of the application.

**Que : 7  How to install React Js on Windows, Linux Operating System? How to Install NPM and How to check version of NPM?**

**Ans.**

**Install React.js and NPM on Windows:**

1. Install Node.js and NPM:

* Visit the official Node.js website: [Node.js Downloads](https://nodejs.org/en/download/).
* Download the latest version of Node.js for Windows.
* Run the installer and follow the installation prompts.

2. Verify Node.js and NPM Installation:

* Open a command prompt or PowerShell.
* Check the installed versions:

        node -v

        npm -v

**Install React.js and NPM on Linux:**

1. Install Node.js and NPM:

* Open a terminal.
* Use the package manager to install Node.js and NPM. The commands might vary based on your Linux distribution. For example, on Debian/Ubuntu, you can use:

      sudo apt update

      sudo apt install nodejs

      sudo apt install npm

2. Verify Node.js and NPM Installation:

* Open a terminal.
* Check the installed versions:

node -v

npm -v

**Installing React.js:**

1. Create a React App:

* Open a terminal or command prompt.
* Use Create React App to generate a new React application. Replace "my-react-app" with your desired project name:

          npx create-react-app my-react-app

2. Navigate to the App Directory:

* Change to the project directory:

             cd my-react-app

3. Start the Development Server:

* Start the development server:

          npm start

* Open your web browser and visit [http://localhost:3000](http://localhost:3000/) to see your React app.

            Checking NPM Version:

* To check the installed version of NPM, use:

          npm -v

This process should help you install React.js and NPM on both Windows and Linux.

**Que : 8  How to check the version of React Js?**

**Ans:**

          To check the version of React we can use these 3 approaches:

1. Check React Version in Package.json:

          Open your project's `package.json` file. This file contains metadata about your project, including the versions of the installed packages. Look for the` "react"` dependency and its version. It should be listed under the   `"dependencies"` or `"devDependencies" `section.

2. Check React Version in the Command Line:

If you want to check the React version directly from the command line, you can use the following command. Open a terminal or command prompt, navigate to your project directory, and run:

    npm list react

3. Check React Version in the Browser Developer Tools:

If your React app is running, you can also check the React version using your browser's developer tools. Open your app in the browser, right-click on the page, select "Inspect" or "Inspect Element," and go to the "Console" tab.Then, enter the following command:

// Filename - App.js

import React from "react";

const App = () => {

    return (

        <h1>

            We are currently using react version{" "}

            {React.version}

        </h1>

    );

};

export default App;

**Que : 9  How to change in components of React Js?**

**Ans.**

          setState() enqueues changes to the component state and tells React that this component and its children need to be re-rendered with the updated state. This is the primary method you use to update the user interface in response to event handlers and server responses.

**Que : 10  How to Create a List View in React Js?**

**Ans.**

**JSX:**

import React, { useState } from "react";

export default function Listview() {

  const [questions, setQuestions] = useState([

    // create state for list view

    {

      id: 1,

      question: "THE 'REACT WAY' TO RENDER LIST ",

      options: [

        "Use Array.map",

        "Not a for loop",

        "Give each item a unique key",

        "Avoid using array index as the key",

      ],

      // correctAnswer: ["Use Array.map"],

      selectedOptions: [], //For select Some options

    },

  ]);

  // For choose options

  const handleOptionSelect = (questionId, option) => {

    setQuestions((prevQuestions) =>

      prevQuestions.map((q) => {

        if (q.id === questionId) {

          const selectedOptions = q.selectedOptions.includes(option)

            ? q.selectedOptions.filter((selected) => selected !== option)

            : [...q.selectedOptions, option];

          return { ...q, selectedOptions };

        } else {

          return q;

        }

      })

    );

  };

  return (

    <>

      <div className="list-view">

        {questions.map((q) => (

          <div key={q.id}>

            <h1 className="Question">{q.question}</h1>

            <ul>

              {q.options.map((option) => (

                <li key={option}>

                  <label className="custom-checkbox-label">

                    {/\* For checkbox \*/}

                    <input

                      className="custom-checkbox-input"

                      type="checkbox"

                      name={`question\_${q.id}`}

                      value={option}

                      checked={q.selectedOptions.includes(option)}

                      onChange={() => handleOptionSelect(q.id, option)}

                    />

                    {/\* for options \*/}

                    <span className="custom-checkbox-text">{option}</span>

                  </label>

                </li>

              ))}

            </ul>

          </div>

        ))}

      </div>

    </>

  );

}

**App.css:**

body {

  background-color: rgb(126, 233, 97);

}

li {

  list-style: none;

}

/\* ------ Listview.css start ------ \*/

.list-view {

  position: relative;

  top: 100px;

}

.custom-checkbox-label {

  display: flex;

  align-items: center;

}

.custom-checkbox-input {

  position: relative;

  left: 30%;

  margin: 5px;

  width: 30px;

  height: 30px;

  border-radius: 50%;

  vertical-align: middle;

  background: gainsboro;

  border: 1px solid gainsboro;

  appearance: none;

  -webkit-appearance: none;

  outline: none;

  cursor: pointer;

}

.custom-checkbox-input:checked {

  appearance: auto;

  clip-path: circle(50% at 50% 50%);

  background-color: blue;

}

.custom-checkbox-text {

  position: relative;

  left: 30%;

  height: 30px;

  width: 400px;

  text-align: left;

  font-size: 18px;

  border: 1px solid #fff;

  padding: 2px 5px;

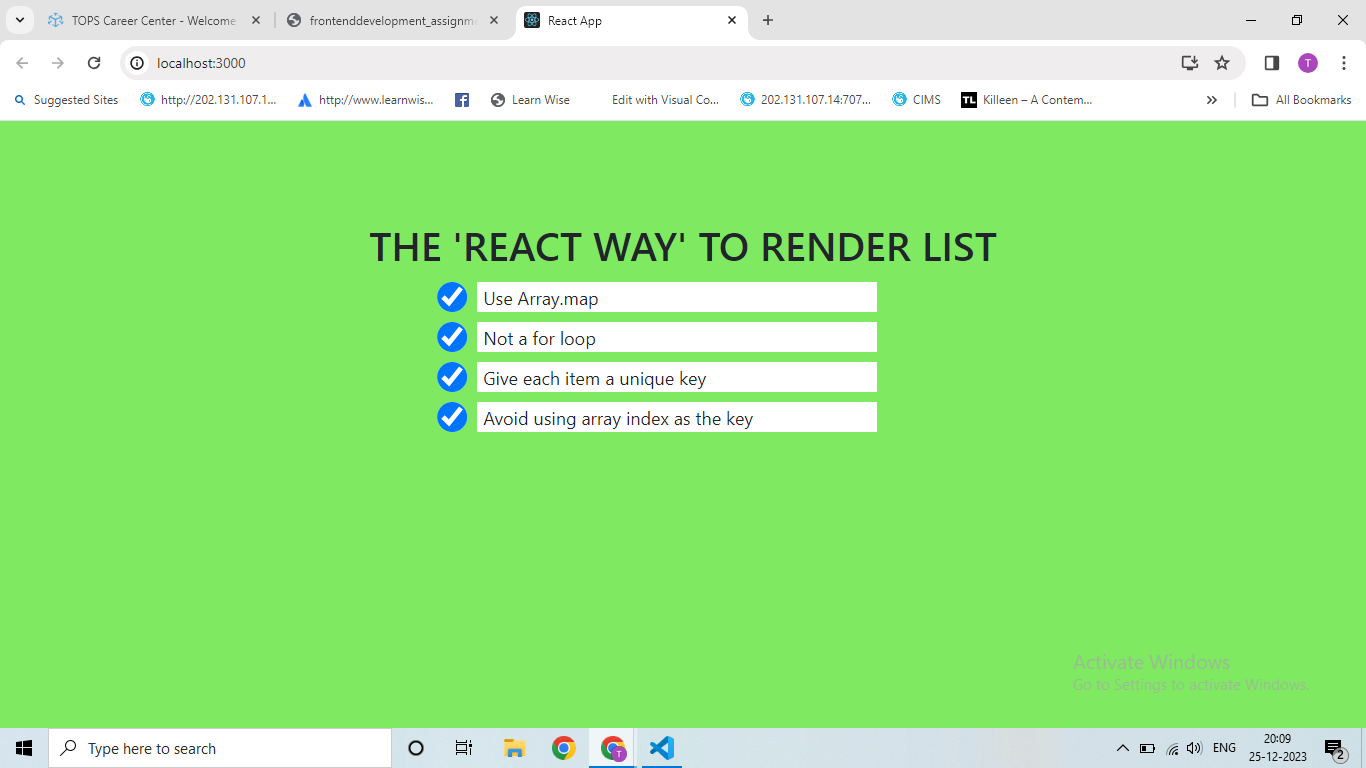
  margin: 5px;

  background: #fff;

}

/\* ------ Listview.css end ------ \*/

**OUTPUT :**



**Que : 11  Create Increment decrement state change by button click?**

**Ans.**

**JSX:**

import { useState } from "react";

import ReactDOM from "react-dom/client";

export default function Increment() {

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

  // Function create for Decrement counting onClick button

  const Decrement = () => {

    if (count > 0) {

      setCount(count - 1);

    }

  };

  // Function to Reset the count to 0

  const handleReset = () => {

    setCount(0);

  };

  return (

    <>

      <div className="counter">

        <h1>React Web</h1>

        <div className="count">{count}</div>

        <button onClick={Decrement}>Decrement</button>

        {/\* Function to Increment counting onClick button \*/}

        <button onClick={() => setCount((I) => I + 1)}>Increment</button> <br />

        <button onClick={handleReset}>Reset</button>

      </div>

    </>

  );

}

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

root.render(<Increment />);

**App.css:**

body {

  background-color: royalblue;

  color: #fff;

}

.counter {

  position: relative;

  top: 150px;

}

h1,

div {

  text-align: center;

}

.count {

  color: rgb(35, 34, 34);

  font-size: 20px;

  font-weight: bolder;

}

button {

  background: rgb(35, 34, 34);

  color: #fff;

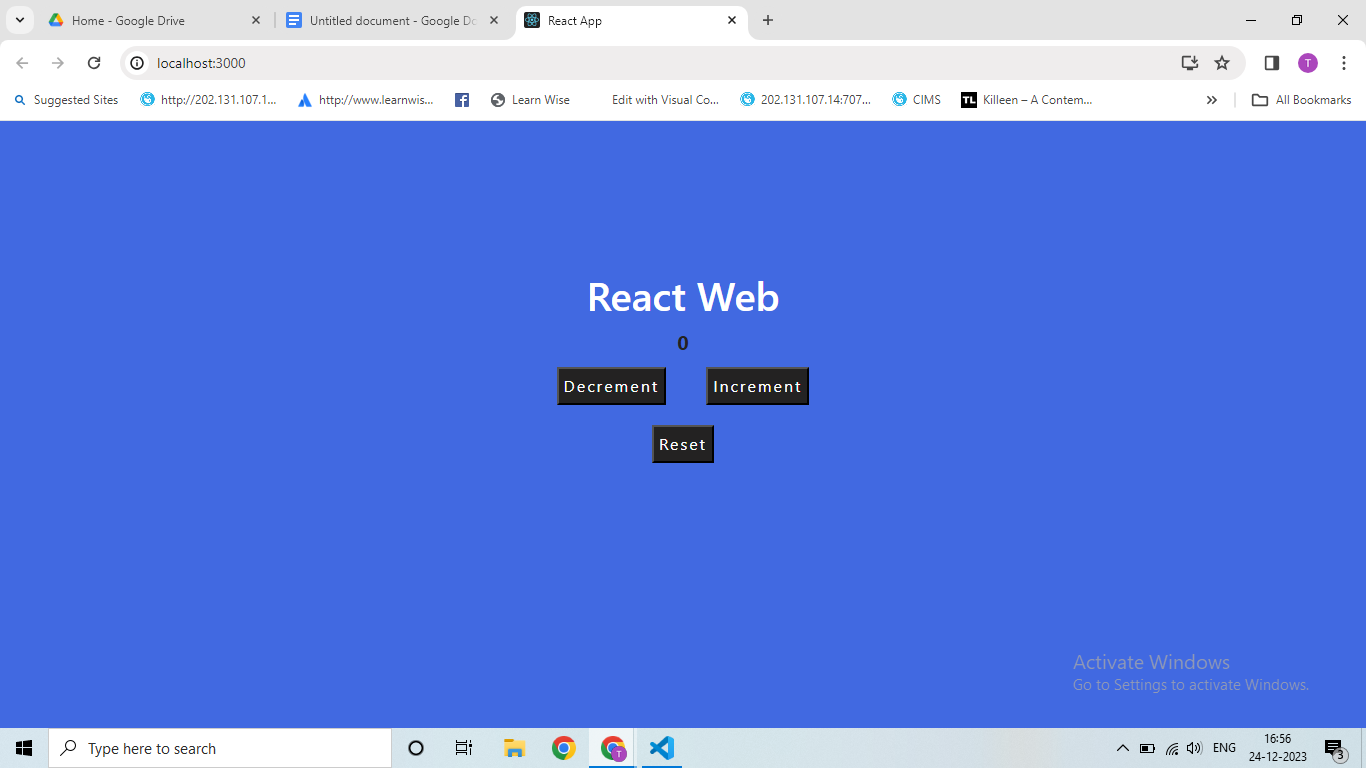
  margin: 10px 20px;

  padding: 5px;

  letter-spacing: 2px;

}

**OUTPUT :**

****