**Node - Node with MongoDB**

**Q1:** **What is MongoDB?**

**Ans: MongoDB** is a popular NoSQL database that stores data in a flexible, document-oriented format. Unlike traditional relational databases (like MySQL or PostgreSQL) that use tables and rows, MongoDB stores data in documents, which are similar to JSON objects.

MongoDB is a versatile and scalable database that is well-suited for a wide range of applications. Its flexible document-oriented model, high performance, and rich feature set make it a popular choice for modern data storage needs.

**Q2:** **What is difference between mongo DB and SQL?**

**Ans: MongoDB** and **SQL** (Structured Query Language) are both popular database systems, but they have distinct approaches to storing and managing data.

**MongoDB (NoSQL):**

* **Document-oriented:** Data is stored in flexible JSON-like documents.
* **Schema-less:** No strict schema definition is required, allowing for dynamic changes.
* **High performance:** Optimized for handling large datasets and complex queries.
* **Scalability:** Easily scales horizontally by adding more servers.
* **Use cases:** Content management systems, real-time analytics, IoT applications.

**SQL (Relational):**

* **Relational:** Data is stored in tables with rows and columns.
* **Schema-defined:** Requires a predefined schema, making it more rigid.
* **Strong data integrity:** Enforces data consistency and integrity through constraints and relationships.
* **Transaction support:** Supports ACID properties (Atomicity, Consistency, Isolation, Durability) for reliable transactions.
* **Use cases:** Business applications, financial systems, e-commerce platforms.

**Q3:** **What is Role of Node JS in react JS?**

**Ans:** Node.js and React.js serve different purposes in web development, and they are often used together to create a full-stack JavaScript application. Here's overview of the role of Node.js in a React.js application:

Node.js is often employed as the backend server for a React application. It can handle API requests, communicate with databases, and perform other server-side operations.

Node.js can be used to implement server-side rendering (SSR) for React applications.

Node.js can be used to connect to databases (such as MongoDB, MySQL, or PostgreSQL) and handle database operations for a React application.

**Q4: What is CLI command In React JS?**

**Ans:** The Command Line Interface (CLI) is a set of tools that helps developers manage React applications more efficiently. The most commonly used CLI tool for React development is create-react-app, which is a popular tool to set up a new React project with a predefined and optimized project structure. Here are some common CLI commands in React:

**Create a New React App:**

To create a new React application, you use the create-react-app command followed by the name of your project. This command sets up a new project with a default folder structure, configuration files, and development dependencies.

**npx create-react-app my-react-app**

**Start Development Server:**

To start the development server and run your React application locally, you use the following command:

npm start

**Q5:** **What is Components in React JS?**

**Ans:** A component is a self-contained, reusable building block that encapsulates a piece of user interface (UI) and its associated behavior. React applications are composed of components, and these components can be nested and combined to create complex UIs. Components are a fundamental concept in React, promoting modularity, reusability, and maintainability in application development.

**There are two types of components in react js**

**1.Function components**

**2.Class components**

**Q6:** **What is Header and Content Components in React Js?**

**Ans:** The terms "Header" and "Content" components are often used to represent different sections or elements of the user interface. These components are typically part of a broader application structure and are responsible for rendering specific parts of the UI. Let's explore the concepts of Header and Content components in more detail:

**Header Component:**

The Header component in a React application is responsible for rendering elements that appear at the top of the user interface. This can include navigation menus, branding elements, user authentication status, and other information that is consistently displayed across different pages or views.

The Header component is often designed to provide a navigation structure, allowing users to move between different sections of the application.

**Example Header component:**

//Header component

function Header () {

return (

<header>

<h1>My React App</h1>

<nav>

<ul>

<li>Home</li>

<li>About</li>

<li>Contact</li>

</ul>

</nav>

</header>

);

}

**Content Component:**

The Content component is responsible for rendering the main content of the application. This can include dynamic data, user-specific information, and the primary features or views of the application.

The Content component is often dynamic and changes based on user interactions or the application's state. It is the central area where the primary functionality and information are displayed.

**Example Content component:**

// Content Component

function Content() {

return (

<div>

<h2>Welcome to the Content Section</h2>

<p>This is the main content of the application.</p>

</div>

);

}

**Q7:** **How to install React Js on Windows, Linux Operating System? How to install NPM and How to check version of NPM?**

**Ans:**

**Installing React.js and npm on Windows:**

**1.Install Node.js:**

Download the latest version of Node.js from the official website: Node.js

Run the installer and follow the on-screen instructions.

Check Node.js and npm Installation:

**2.Open a command prompt or PowerShell window.**

**To check if Node.js is installed, run:**

**node --version**

To check if npm is installed, run:

**npm --version**

**3.Install Create React App:**

To install Create React App, a tool that simplifies the React project setup, run:

**npm install -g create-react-app**

Create a React App:

Once installed, you can create a new React app using Create React App:

**npx create-react-app my-react-app**

Navigate to the App Directory:

Change into the newly created app directory:

**cd my-react-app**

**4.Start the Development Server:**

Start the development server to run your React app:

**npm start**

**Access your app at http://localhost:3000 in a web browser.**

**Installing React.js and npm on Linux:**

1. **Install Node.js:**

**Open a terminal window.**

Run the following commands to install Node.js:

**sudo apt update**

**sudo apt install nodejs**

**sudo apt install npm**

1. **Check Node.js and npm Installation:**

To check if Node.js is installed, run:

**node -version**

To check if npm is installed, run:

**npm -v**

1. **Install Create React App:**

**To install Create React App, run:**

**sudo npm install -g create-react-app**

1. **Create a React App:**

Once installed, you can create a new React app using Create React App:

**npx create-react-app my-react-app**

**Navigate to the App Directory:**

Change into the newly created app directory:

**cd my-react-app**

1. **Start the Development Server:**

Start the development server to run your React app:

**npm start**

**Access your app at http://localhost:3000 in a web browser.**

**Q8:** **How to check version of React Js?**

**Ans:** To check the version of React.js that is installed in your project, you can look at the **package.json** file, which is located in the root directory of your React application. The **package.json** file includes information about your project, including the versions of the dependencies, including React.

**Here are the steps to check the version of React.js in your project:**

1.Open a command prompt, PowerShell, or terminal window.

2.Navigate to the root directory of your React project.

3.Open the package.json file in a text editor. You can use a code editor like Visual Studio Code, Atom, or any text editor of your choice.

4.Look for the "dependencies" section in the package.json file. It should include an entry for React, and the version number will be specified.

**"dependencies": {**

**"react": "^17.0.2",**

**"react-dom": "^17.0.2",**

**}**

**Q9:**  **How to change in components of React Js?**

**Ans:** Three ways to make a change in component:

**1.Update in State**

* The state change can be from a prop or setState change to update a variable.
* The component gets the updated state and React re-renders the component to reflect the change on the app.

**2.Update in prop :**

* Likewise the change in prop leads to state change and state change leads to re-rendering of the component by React.

**3.Re-rendering of parent component :**

* Whenever the components render function is called, all its subsequent child components will re-render, regardless of whether their props have changed or not.

**Q10:** **How to Create a List View in React JS ?**

**Ans:**

**Step 1:**

* Create a list of elements in React in the form of an array ans store it in a variable.
* We will render this list as an unordered list element in the browser.

**Step 2:**

* We will then traverse the list using the JavaScript map() function and updates elements to be enclosed between <li> </li> elements.

**Step 3:**

* Finally we will wrap this new list within <ul> </ul> elements and render it to the DOM

## Example:

## Import React from ‘react’;

## Import ReactDOM from ‘react-dom’;

## const numbers = [ 1,2,3,4,5 ];

## const updateNums = number.map((number) => {

## return <li> {number} </li>;

## });

## ReactDOM.render(

## <ul>

## { updateNums };

## </ul>,

## document.getElementByID(‘root’)

## );

**Q11:** **Create Increment decrement state change by button click?**

**Ans:** import React, { useState } *from* "react";

import "./IncrementDecrementBtn.css";

const IncrementDecrementBtn = ({ minValue = 0, maxValue = 100 }) => {

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

    const handleIncrementCounter = () => {

      if (count < maxValue) {

        setCount((prevState) => prevState + 1);

      }

    };

    const handleDecrementCounter = () => {

      if (count > minValue) {

        setCount((prevState) => prevState - 1);

      }

    };

    return (

      <div *className*="btn-group">

        <button *className*="increment-btn" *onClick*={handleIncrementCounter}>

          <span *class*="material-symbols-outlined">Increment</span>

        </button>

        <p>{count}</p>

        <button *className*="decrement-btn" *onClick*={handleDecrementCounter}>

          <span *class*="material-symbols-outlined">Decrement</span>

        </button>

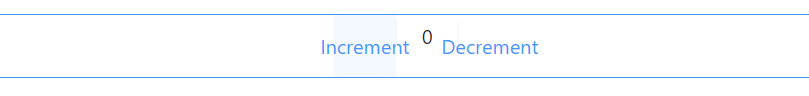
      </div>

    );

  };

export default IncrementDecrementBtn

**OUTPUT:**

****