1. Introduction to React
2. React Setup
3. JSX (JavaScript XML)
4. Components
   * Functional Components
   * Class Components
   * Props
   * State
5. React Hooks
   * useState
   * useEffect
   * Custom Hooks
6. Event Handling
7. Lists and Keys
8. Forms in React
   * Controlled Components
   * Uncontrolled Components
9. React Router
10. State Management

* Lifting State Up
* Context API

1. React Lifecycle Methods (Class Components)
2. React with APIs
3. Deployment
4. React Best Practices

 React js and course Introduction

 React js with CDN and without any installation

 Installation for React js (node, npm, create-react-app) on Windows OS

 Write First Code with ReactJs

 React js project File and Folder Structure

 What is package.json in React js

 Component in React

 Class component in React js

 What is JSX in React

 Click event in React js

 State in React functional component

 State in class component in React

 Props in the functional component in React

 Props in class component

 Get input box value

 Show and hide element (toggle)

 Basic form tutorial

 If-else conditions

 Pass function as props

 Introduction to React js lifecycle

 Constructor lifecycle method

 Render lifecycle in React js

 Component did mount React js lifecycle

 Component did update React js lifecycle

 Should Component update in React js lifecycle

 ComponentWillUnmount in React js

 Hooks in React js

 useEffect hook in React js

 useEffect with state and props in React js

 Style types in React js

 Install Bootstrap in React js

 Handle array with list in React js

 List with Bootstrap in React js

 Nested list with a nested array in React tutorial

 How to reuse components in React js

 Nested list with a nested array in React tutorial

 React js send data from child to parent

 Pure Component in React js

 Use memo in React functional component

 Ref in React js

 useRef in React functional component

 React js forward ref

 React js Controlled Component

 React js Uncontrolled Component

 React js Higher-order component

 React js Routing basic

 React js Routing example

 404 page in React routing

 React js Dynamic routing

 React js Call Get Method API - HTTP call

 What is Postman

 How to call Post method API in React js

 React js delete method API

 React js pre-filled form

 React js update API method

 React js get the previous state in the functional component

 Get previous props in React js with hooks

 Context API in React js

 React js interview Questions

**Day 1: Introduction and Setup**

* Introduction to React
* Setting up React environment (Create React App)

**Day 2: JSX (JavaScript XML)**

* Understanding JSX
* Embedding expressions in JSX
* Attributes in JSX

**Day 3: Components**

* Functional Components
* Class Components

**Day 4: Props and State**

* Passing Props
* Managing State in Class Components
* useState Hook in Functional Components

**Day 5: Handling Events**

* Handling events in React
* Synthetic events
* Passing arguments to event handlers

**Day 6: Conditional Rendering**

* Conditional rendering using if-else
* Conditional rendering using ternary operators

**Day 7: Lists and Keys**

* Rendering lists
* Using keys in lists

**Day 8: Forms and Controlled Components**

* Handling forms in React
* Controlled components
* Handling multiple inputs

**Day 9: Lifecycle Methods**

* Component lifecycle methods (Class components)
* useEffect Hook

**Day 10: Context API**

* Introduction to Context API
* Using Context for state management

**Day 11: React Router**

* Introduction to React Router
* Setting up routing in a React application
* Navigating between routes

**Day 12: Redux (State Management)**

* Introduction to Redux
* Setting up Redux in a React application
* Basic concepts: Actions, Reducers, Store

**Day 13: Advanced Redux**

* Advanced Redux concepts: Middleware, Thunks
* Using Redux for async actions

**Day 14: Styling in React**

* CSS in React
* CSS-in-JS libraries (styled-components, Emotion)
* Modular CSS

**Day 15: Project Day**

* Building a small project to practice the concepts learned
* Reviewing and refining the project

**Day 1: Introduction and Setup**

* Introduction to React
* Setting up React environment (Create React App)

## **Introduction to React**

### **What is React?**

React is a **JavaScript library** developed by **Meta (formerly Facebook)** for building **user interfaces** (UIs). It allows developers to create dynamic and interactive UIs with reusable components.

#### **Key Features of React:**

1. **Component-Based Architecture**:
   * React is built around components. Components are the building blocks of a React application.
   * Each component is a piece of the UI, and each component can have its own state and logic.
2. **Declarative Programming**:
   * With React, you describe how your UI should look at any point in time, and React will update and render the components when the state changes.

(  
**Declarative Programming** ek programming style hai jisme aap ye nahi batate ki kaise koi kaam hona chahiye, balki aap sirf ye batate hain ki kaunsa kaam hona chahiye. Matlab, aap logic ko express karte hain bina step-by-step instructions diye.

Isme aap "what" ko define karte hain, na ki "how"

)

### **Why Use React?**

1. **Reusable Components**: React allows you to create components that can be reused across the application.
2. **High Performance**: The Virtual DOM allows React to perform updates efficiently.
3. **Strong Ecosystem**: React has a large and active community with a wealth of libraries, tools, and resources.
4. **Declarative**: React’s declarative nature makes code easier to read and maintain.
5. **Learn Once, Write Anywhere**: React can be used for web development, mobile apps (React Native), and more.

## **Setting up React Environment (Create React App)**

### **What is Create React App?**

**Create React App (CRA)** is an officially supported way to create React applications. It sets up everything you need to get started with React, including Webpack, Babel, ESLint, and more. It saves you from having to manually configure these tools.

### **Steps to Set Up a React Environment**

#### 1. **Install Node.js and npm**

node -v   
npm –v

#### 2. **Create a New React App Using Create React App**

* Open your terminal or command prompt.
* Run the following command to create a new React application:

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

This command will:

* + Create a new directory (my-first-react-app).
  + Install all dependencies (React, ReactDOM, etc.).
  + Set up a development environment

#### 3. **Navigate to Your Project Directory**

* Move to your project directory:

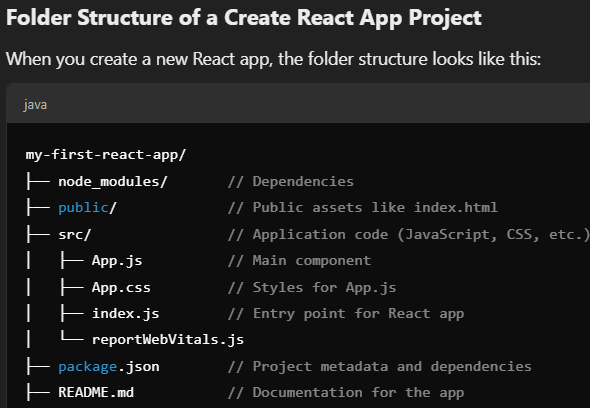
**cd my-first-react-app**

#### 4. **Start the Development Server**

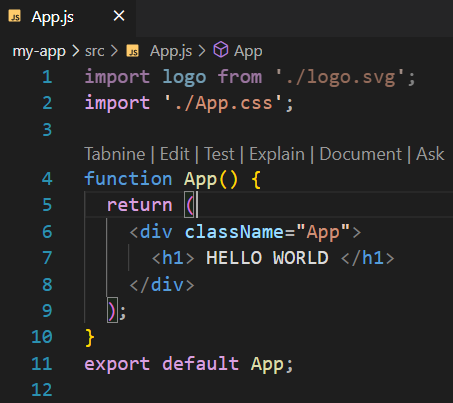
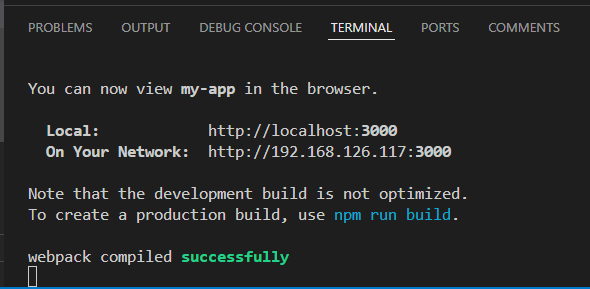
* To view your app in the browser, start the development server:

**npm start**

**This will automatically open the app in your default browser at 🡪 http://localhost:3000.**

**Important Files and Folders**:

* **public/index.html**: The HTML template that gets loaded into the browser.
* **src/index.js**: The entry point of the app where ReactDOM renders the App component.
* **src/App.js**: The main React component that gets rendered by index.js.
* **package.json**: The file that contains project metadata and dependencies.

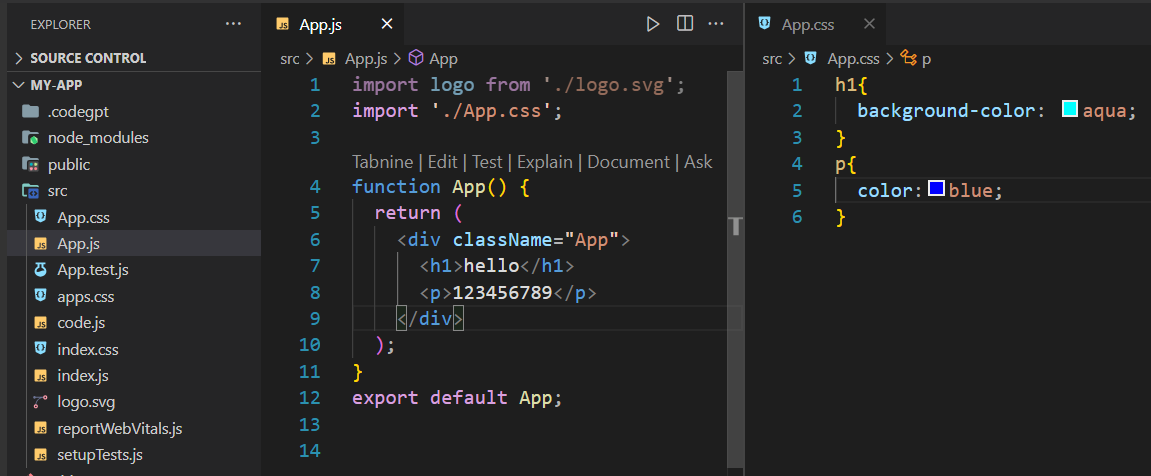
**Day 2: JSX (JavaScript XML)**

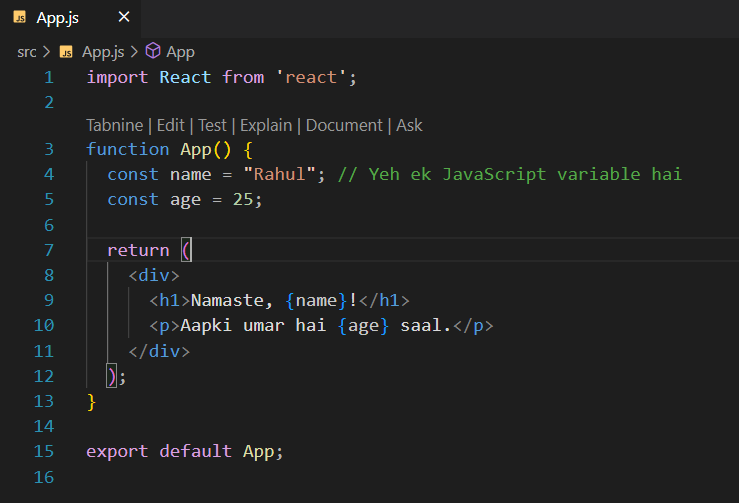
* Understanding JSX
* Embedding expressions in JSX
* Attributes in JSX

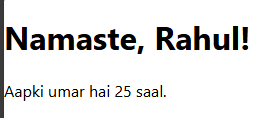
JSX (JavaScript XML) is a syntax extension for JavaScript, which allows you to write HTML directly within JavaScript code. It makes it easier to create React components by describing what the UI should look like in a declarative way. JSX is not a necessity for using React but it makes the code easier to understand and maintain.

#### Key Points:

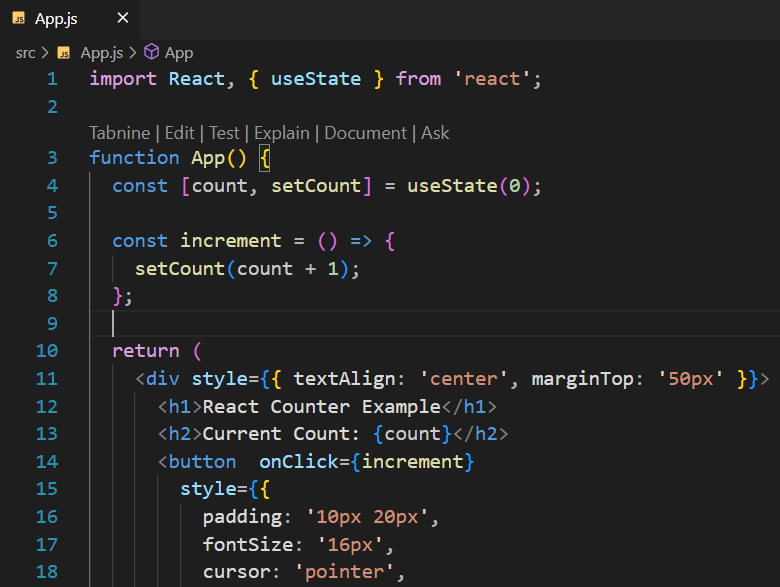
* **Looks like HTML:** JSX closely resembles HTML, but it has the full power of JavaScript.
* **Transpiled to JavaScript:** Browsers don’t understand JSX natively, so it gets transpiled into regular JavaScript by tools like Babel.
* **React Elements:** JSX produces React elements which are the building blocks of React applications.

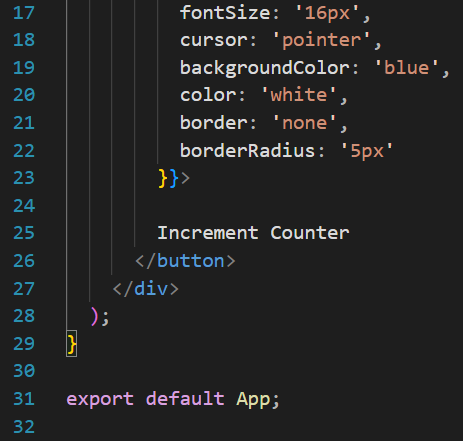


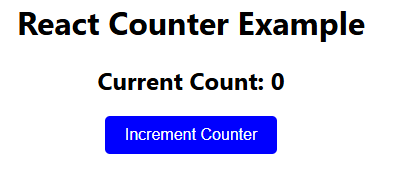




**Example 🡪**







* + 1. **TASK**

**ADD DECREMENT BUTTON AND RED COLOR**

### 🡪 1. **useState Hook**: State kya hoti hai?

* React me **state** ek variable ki tarah hoti hai jo kisi component ka data store karne ke kaam aati hai.
* **useState** ek function hai jo React me state banane aur usse update karne ke liye use hota hai.

#### Example:

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

Yahaan:

 count ek **state variable** hai jisme counter ka current value store hota hai.

 useState(0) ka matlab hai initial value 0 set karna.

*  setCount ek function hai jo count ka value badalne ke liye use hota hai.
* **State variable** ek aisa variable hai jo **React component ke andar data store karne ke liye** use hota hai. State ka kaam hai **component ke behavior** aur **UI ko dynamic banana**.

### 2. **Event Handling**: Button se function call kaise hota hai?

Jab tum kisi button ya kisi HTML element ke saath kaam karte ho, to React me events ko handle karte hain.

#### Example:

**<button onClick={increment}>Increment Counter</button>**

onClick: Jab user button pe click karega, tab React increment function ko call karega.

Aur **increment** function me:

**const increment = () => {**

**setCount(count + 1);**

**};**

* setCount(count + 1) ka matlab hai:
  + **Jo current value hai, usme 1 add karo.**
  + State ko nayi value ke saath update kar do.

### 3. **Dynamic Rendering**: Screen pe value kaise update hoti hai?

Jab tum state (count) ko update karte ho, React automatically screen ko update kar deta hai. Isse **Dynamic Rendering** kehte hain.

#### Example:

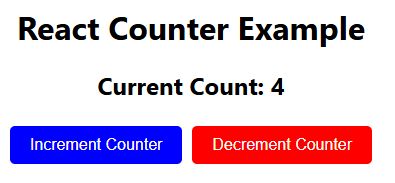
**<h2>Current Count: {count}</h2>**

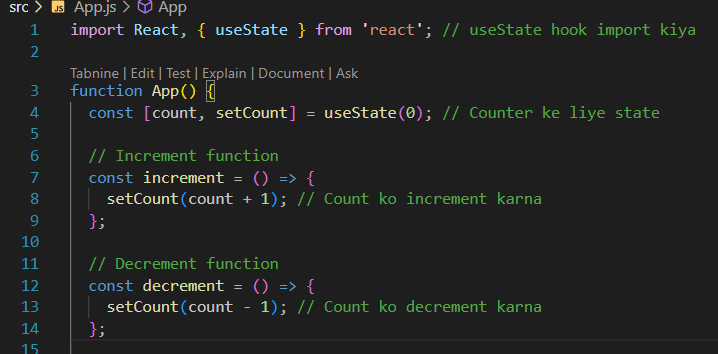
 **State (useState)**: Yeh number (count) tumhare paas ek box me stored hai.

 **Event Handling (onClick)**: Jab tum ek button dabate ho, board ke paas ek banda jata hai aur us box (state) me number badhakar naya number likh deta hai.

 **Dynamic Rendering**: Banda jaise hi board pe number change karta hai, tumhe turant updated board dikhayi deta hai.

Task 1 🡪 o/p



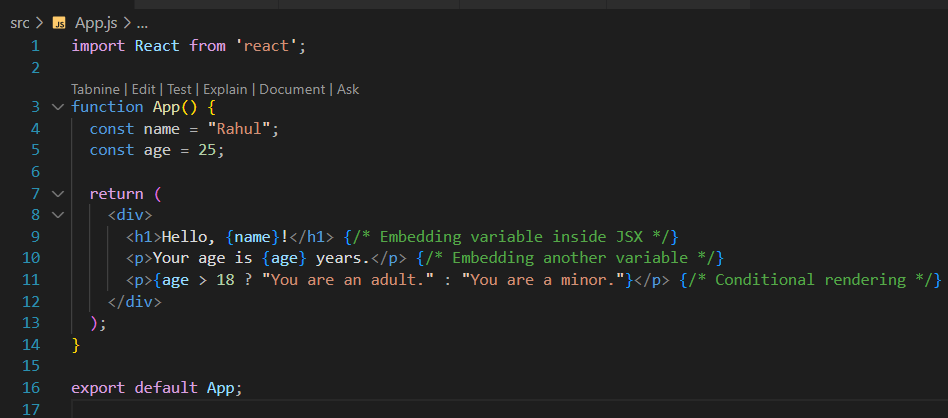






### 1. **Embedding Expressions in JSX**:

JSX mein **expressions** ko curly braces {} ke andar likh kar embed kiya jaata hai. Expressions ka matlab hai koi bhi JavaScript code jo koi value return kare, jaise variables, functions, operations, etc.



### Explanation:

* {name}: Variable name ko JSX ke andar embed kiya gaya.
* {age}: Variable age ko JSX mein display kiya gaya.
* {age > 18 ? "You are an adult." : "You are a minor."}:
* Ek **conditional expression** ko embed kiya gaya, jisme age ke basis par text change hota hai.

### 2. **Attributes in JSX**:

JSX mein HTML ki tarah **attributes** use kiye jaate hain, lekin kuch differences hain.

* **CamelCase**: React mein kuch attributes ko camelCase format mein likhna padta hai, jaise class ko className likhna padta hai.
* **Strings**: JSX mein attributes ke values ko **quotes** mein wrap kiya jaata hai, jaise normal HTML mein hota hai.
* **Boolean attributes**: React mein attributes jo boolean values hold karte hain, unhe bas likhna hota hai bina koi value ke.

