REACT JS

# INRODUCTION

React is a powerful JavaScript library for building user interfaces, especially for single-page applications (SPAs). Developed by Facebook, it allows developers to create reusable UI components that manage their own state and can efficiently update and render the right components when data changes.

# Key Concepts to Get Started:

1. **Components:** The building blocks of any React application. A component is a JavaScript function or class that optionally accepts inputs (called props) and returns a React element that describes how a section of the UI should appear.
2. **JSX:** JavaScript XML, a syntax extension that allows you to write HTML elements directly within JavaScript. It is transformed into regular JavaScript at runtime.
3. **Props (Properties):** Props are read-only inputs to components. They allow passing data from a parent component to a child component.
4. **State:** A way to store data that changes over time and affect what gets rendered. Unlike props, state is managed inside the component.
5. **Rendering:** The process where React updates the DOM to reflect changes in the component's state or props.

# Setting Up React

To start a React project, you'll use a tool called `create-react-app`:

* **Install Node.js (if not already installed):**

Download from [Node.js official website](https://nodejs.org/).

* **Create a New React Project:**

Open your terminal and run the following commands:

npm create vite@latest

cd my-project

npm install

npm run dev

This will start the development server and open the React app in the browser.

# 

# Folder Structure

* A typical Vite + React project structure looks like this:



* The `src` folder contains your React components and application logic, while the `public` folder holds static files.

# Workflow of React

* **Development:** Write components in the `src` directory, manage state and props, and handle events.
* **Rendering:** Components are rendered to the DOM based on their state and props.
* **Hot Module Replacement**: Vite supports hot reloading, allowing you to see changes in real-time without refreshing the browser.

# What is a Component?

* Components are reusable pieces of UI that are either functional or class-based. They accept inputs (props) and return React elements that define how a section of the UI should appear.

# Functional vs Class Component

* **Functional Components:** Simple functions that return JSX. They can use hooks to manage state and lifecycle methods.

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

* **Class Components:** ES6 classes that extend `React.Component` and can hold state and lifecycle methods. However, they are less common with the introduction of hooks.

class Greeting extends React.Component {

render() {

return <h1>Hello, {this.props.name}!</h1>;

}

}

* Recommendation: Prefer functional components for new code, as they are simpler and more concise.

# Functional Components

## Props

* Props are the mechanism for passing data from a parent component to a child component.
* Example:

function UserProfile({ name, age }) {

return (

<div>

<h2>Name: {name}</h2>

<p>Age: {age}</p>

</div>

);

}

## Props Children

* Special prop called `children` allows components to pass child elements:

function Container({ children }) {

return <div className="container">{children}</div>;

}

// Usage

<Container>

<h1>Hello</h1>

<p>This is a child element!</p>

</Container>

## Props as Method

* Props can be functions passed down to child components:

function ParentComponent() {

const handleClick = () => {

alert("Button clicked!");

};

return <ChildComponent onClick={handleClick} />;

}

function ChildComponent({ onClick }) {

return <button onClick={onClick}>Click Me</button>;

}

## State

* State is a way to store data that can change over time and affect what is rendered. In functional components, state is managed using the `useState` hook:

import { useState } from 'react';

function Counter() {

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

return (

<div>

<p>You clicked {count} times</p>

<button onClick={() => setCount(count + 1)}>Click me</button>

</div>

);

}