LESSON 03
Introduction to React & Project Setup

WEEK 01

Objectives

- Understanding React Fundamentals
- Setting Up Development Tools
- Setting up a React Project
- React Project structure
- React Examples

Understanding React Fundamentals

What is React?

- JavaScript library for building user interfaces.
- Developed by Facebook, open-source since 2013.
- Focuses on component-based architecture.

Key Features

- **JSX**: JavaScript XML syntax for describing UI
- **Components**: Reusable, modular UI pieces
- **Virtual DOM**: Efficient rendering with minimal updates
- **One-way Data Flow**: Predictable state management

Why Use React?

- ❖ Fast and responsive UIs
- ❖ Simplified development with reusable components
- ❖ Large ecosystem and community support
- ❖ Works with modern frameworks and tools

React Core Concepts

❖ Components

- ➢ Modular UI building blocks
- ➢ Functional or Class-based
- ➢ Reusable and composable

React Core Concepts

❖ JSX

- ➢ JavaScript XML syntax
- ➢ Combines HTML-like code with JavaScript
- ➢ Transpiled to pure JavaScript
- ➢ Enables dynamic UI

React Core Concepts

❖ Virtual DOM

- ➢ Lightweight in-memory representation of real DOM
- ➢ Optimizes UI updates
- ➢ Improves performance

React Core Concepts

- **Props:** (short for properties) are used to pass data from parent components to child components.
  - Data passed to components
  - Immutable within component
  - Enables customization

React Core Concepts

- **State:** State is an object that holds data or information about the component and determines how it renders and behaves
  - Internal component data
  - Managed with **useState** hook
  - Triggers re-renders

React Core Concepts

❖ Component Lifecycle

The lifecycle of a React component refers to the series of methods that are invoked in different phases: mounting, updating, and unmounting.

❖ Common Lifecycle Methods

componentDidMount(), **componentDidUpdate()**, and **componentWillUnmount()** are common lifecycle methods used to perform actions at specific points.

React Core Concepts

- **Hooks**
  - Functions to add state and lifecycle to functional components
  - useState, useEffect, useContext, etc.

React Core Concepts

❖ Event Handling in React

- ➢ React uses synthetic events for consistent browser behavior
- ➢ Events are handled via event handlers in components
- ➢ Similar to DOM events but with React-specific syntax

```
1  function Button() {
2    const handleClick = () ⇒ {
3      alert('Clicked!');
4    };
5
6    return <button onClick={handleClick}>Click Me</button>;
7  }
```

Setting Up Development Tools

Install Node.js

- Node.js is required to run JavaScript and manage React dependencies using npm.
- Download from: Node.js Official Site.

•Download from: Visual Studio Code Official Site.

Install Visual Studio Code (VS Code)

- VS Code is a powerful and lightweight code editor used for React development.
- Download from: Visual Studio Code Official Site.

VS Code Extensions for React Development

- **ES7 React/Redux/GraphQL/React-Native Snippets**

  Provides React snippets for faster development of React components and Redux actions.

- **Prettier**

  Automatically formats your code to maintain a consistent style.

- **ESLint**

  Helps in maintaining code quality by identifying and fixing code issues.

Setting up a React Project

Prerequisites

- Install Node.js
  - Ensure you have Node.js installed (version LTS).
- Install npm
  - npm comes bundled with Node.js, which is necessary for installing dependencies.

Setting Up Vite for React Project

❖ What is Vite?

- ➢ Vite is a modern, fast build tool that allows quick setup and development of React applications. It provides fast hot module replacement and optimized builds.

❖ Why Use Vite for React?

- ➢ Vite offers faster startup times and faster updates during development compared to other tools like Create React App (CRA).

Setting Up Vite for React Project

❖ Install Vite

- ➢ To set up a React project using Vite, run the following command:

  npm create vite@latest my-react-app --template react
- ➢ This will scaffold a new React project with Vite.

❖ Install Dependencies

- ➢ Navigate to the project folder and install dependencies:

  cd my-react-app

  npm install

Running the React Project

❖ Start the Development Server

- ➢ Run the project locally with: npm run dev
- ➢ This starts the development server and opens the project in your browser at **http://localhost:5173**

❖ Fast Reload

- ➢ Vite enables hot module replacement (HMR), so changes in your code automatically reload in the browser.

React Project Structure

React Project Structure

❖ Overview of React Project Structure

- ➢ A typical React project has a predefined structure that helps organize components, assets, and configuration files.
- ➢ Common structure includes folders for source code (src), assets, configuration files, and dependencies.

❖ Project Root Directory

- ➢ Contains essential configuration files and scripts to run the app, like package.json, vite.config.js (for Vite), and others like README.md.

React Project Structure

- **node_modules/**
  - Stores all project dependencies
  - Installed via npm/yarn
  - Not tracked in version control
- **public/**
  - Static assets (e.g., images, favicon)
  - index.html: Main HTML file
  - Accessible without bundling
- **src/**
  - Core source code folder
  - Contains components, styles, and logic
  - index.js: Entry point for React app

React Project Structure

- **package.json**
  - Project metadata and scripts
  - Lists dependencies and devDependencies
  - Defines commands (e.g., start, build)
- **.gitignore**
  - Specifies files/folders to ignore in Git
  - Typically includes node_modules/, build/
  - Ensures clean repository
- **README.md**
  - Project documentation
  - Includes setup and usage instructions
  - Written in Markdown

React Examples

React Examples

❖ Counter App

- ➢ A simple app that allows users to increase or decrease a counter value by clicking buttons.

❖ Picture Viewer App

- ➢ A simple app that allows users to view and navigate through a collection of images.

❖ Todo List App Example

- ➢ A simple app where users can add, remove, and mark tasks as completed.

React Examples

❖ Todo List Demo

- ➢ Manages list of tasks
- ➢ Uses useState for tasks, useEffect for storage
- ➢ Shows component composition