WEB TECHNOLOGIES

ReactJS - MERN Introduction

Prof. Pavan A C

Department of Computer Science and Engineering

Acknowledgement:

Teaching Assistants(Harini B and Chandana M S)

Web Development Stack

The Big Picture

Definition: A collection of technologies (frontend, backend, database) working together to build modern web apps.

Fun Analogy: Think of a web dev stack as a **restaurant setup**:

- **Frontend (UI)** = the **menu & dining area** (what customers see).
- **Backend (Server)** = the **kitchen** (where the magic happens).
- **Database** = the **fridge/pantry** (where ingredients/data are stored).

Just like a restaurant needs all 3 to serve food, a web app needs all 3 to serve functionality.

Web Development Stack

Full Web Development Stack

A **full stack** = covering **frontend + backend + database**.

Popular stacks:

- MERN Stack
- MEAN Stack
- MEVN Stack
- Serverless Stack
- LAMP Stack
- PERN Stack
- Ruby on Rails Stack

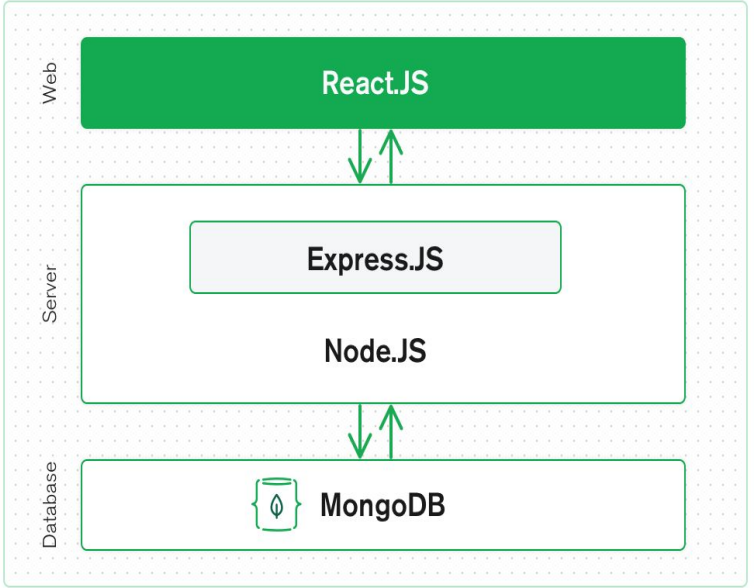
Analogy continuation: A **full restaurant team** — waiters (frontend), chefs (backend), and pantry managers (database). Without one, the restaurant breaks down.

Web Development Stack

MERN Stack Introduction

MERN = MongoDB + Express.js + React.js + Node.js.

- MongoDB - document database
- Express(.js) - Node.js web framework
- React(.js) - a client-side JavaScript framework
- Node(.js) - the premier JavaScrip



100% JavaScript across frontend, backend, and database communication.

Analogy: Imagine running a restaurant where **everyone speaks the same language (say, English)** — smooth communication, fewer errors. That's MERN for devs.

ReactJS (The Frontend Library)

- Top tier of MERN stack.
- A **component-based JS library** for building UIs.
- Handles dynamic, data-driven views efficiently with Virtual DOM.
- **Analogy**: React is the **waiter** — presenting dishes (data) nicely to customers (users), updating the table whenever the order changes.

NodeJS (The Runtime Environment)

- Middle tier of MERN stack.
- A **JavaScript runtime** that runs code on the server side.
- Powers Express, handles requests, connects to DB, runs logic.
- **Analogy**: Node.js is the **kitchen** itself — the environment where cooking (execution) happens. Without the kitchen, the chef can’t cook.

ExpressJS (The Server Framework)

- Middle tier of MERN stack.
- A lightweight **NodeJS framework** for handling routes & HTTP requests.
- Simplifies server logic with middleware & routing.
- **Analogy**: Express is the **chef assistant** — it makes sure the kitchen (Node) follows recipes correctly and serves dishes quickly.

MongoDB (The Database)

- Bottom tier of MERN stack.
- A **NoSQL database** storing data as documents (JSON-like).
- Flexible, schema-less, great for dynamic apps.
- **Analogy**: MongoDB is the **pantry/fridge** of the restaurant, it doesn't enforce rigid "shelves," you can store ingredients in whatever containers you like.

ReactJS

Deep dive

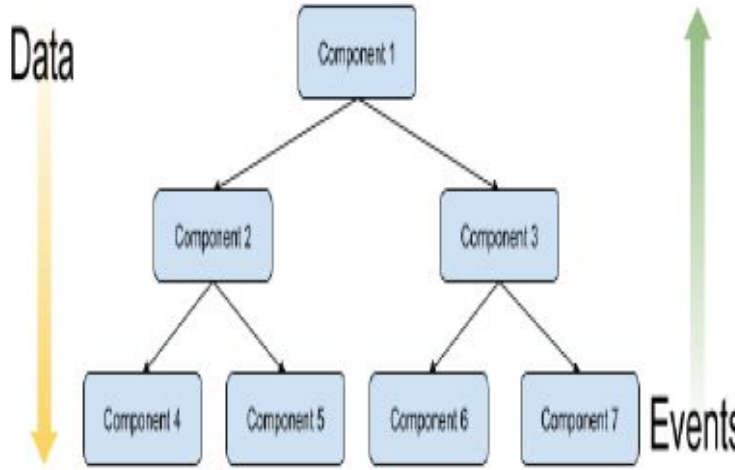
- **What**: Open-source JavaScript library created by Facebook in 2013.
- **Why**: Enables building **single-page apps (spa)** with fast, dynamic updates, without the need of reloading.
- **Core Idea**: UI = components + state + props.
- **Fun Angle**: React is like **LEGO blocks** — build small reusable pieces (components), combine them into big structures (UIs).

Properties of React - A quick rundown

- **Declarative** → You tell React *what* you want, it figures out *how*.
- **Simple** → Clear, focused, readable.
- **Component-based** → Everything reusable, modular.
- **Supports server side** → Rendering can also happen on the server.
- **Mobile support** → Via React Native.
- **Extensive** → Huge ecosystem & libraries.
- **Fast** → Optimized updates via Virtual DOM.
- **Easy to learn** → JavaScript + JSX = beginner-friendly.
- **Single-way data flow** → Props down, actions up.
- **Virtual DOM** → Efficient rendering by comparing virtual vs real DOM.

Single Way Data Flow

Definition: React enforces a **unidirectional data flow**, meaning data always flows from parent → child components.



Mechanism:

- **Props**: Passed down from parent to child (read-only).
- **Callbacks / State Updaters**: Child can communicate changes *up* by invoking functions provided by the parent.
- Actions flow up and properties flow down

Single Way Data Flow

Why this matters:

- Predictable data handling — you always know the source of truth.
- Easier debugging — data changes can be traced step by step.
- Better maintainability — prevents unexpected side-effects from two-way binding.

Analogy: Like a **waterfall** — water (data) flows down from the top (parent), but if something at the bottom (child) wants to influence the top, it sends a **signal** (callback) back up.

Virtual DOM

Definition: The **Virtual DOM (VDOM)** is an in-memory lightweight copy of the actual DOM. React updates this virtual representation first before syncing it with the real DOM.

How it works (Reconciliation Process):

1. React creates a **virtual DOM tree** when you write JSX.
2. On state/prop change, React generates a **new VDOM tree**.
3. React performs a **diffing algorithm** to compare old vs new VDOM.
4. Only the **minimal set of changes** are applied to the real DOM.

Virtual DOM

Why this matters:

- The real DOM is **expensive** to manipulate — each change can trigger re-layouts, re-paints, and re-flows.
- The VDOM reduces these costs by batching & optimizing updates.
- Results in **better performance** and a smoother user experience.

Analogy:

Imagine redesigning your **room**:

- Instead of tearing everything down, you **sketch changes on paper first** (Virtual DOM).
- Once you know what's different (diffing), you only **move the table** or **change the poster**, instead of repainting the entire room (real DOM)

How to Create a React App

Way 1: Using Node Package Manager (npm)

```
npx create-react-app my-app

cd my-app

npm start
```

Way 2: Directly import React in HTML

```
<script src="https://unpkg.com/react@18/umd/react.development.js"></script>

<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```

Docs: [React Official Docs – Getting Started]

MCQs

Q1. In React, data flows in which direction by default?

a) Two-way binding

b) Upwards only

c) Single-way (Parent → Child)

d) Random

Answer: c) Single-way (Parent → Child)

Q2. In the MERN stack, which component is responsible for serving as the runtime environment?

a) MongoDB

b) Express.js

c) React.js

d) Node.js

Answer: d) Node.js

MCQs

Q3. Which feature of React ensures faster rendering by minimizing real DOM updates?

a) Babel

b) JSX

c) Virtual DOM

d) Components

Answer: c) Virtual DOM

References

- React Official Docs – react.dev
- MDN Web Docs – Getting started with React
- W3Schools React Intro
- MongoDB MERN Overview
- GeeksforGeeks – MERN Stack
- Contentful Blog – MERN Stack
- Oracle – MERN vs Other Stacks

THANK YOU

Prof. Pavan A C

Department of Computer Science and Engineering