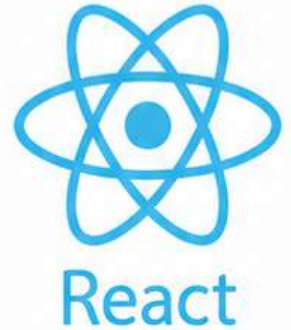


Chapter 1: ReactJS Introduction

1.1 What is React?

- * React is a popular **JavaScript library** for building user interfaces, especially **single-page applications**.
- * It is developed and maintained by **Facebook**.
- * React allows us to create **reusable UI components**.
- * It uses a **component-based architecture**.
- * React makes it easy to build **fast, interactive** and **scalable** web applications.



Key Points :

- ☒ React is a JavaScript library, not a framework.
- ☒ It is used to build UI components.
- ☒ It follows the component-based approach.



1.2 Why React?

- * React provides many advantages that make development easier and faster.
- * It helps developers build large scale applications efficiently.
- * It is **flexible, fast** and has a **large community support**.



- **Component-Based**
Build encapsulated components that manage their own state.



- **High Performance**
Uses Virtual DOM to update only the changed components.



- **Reusability**
Components can be reused in multiple parts of the application.



- **Easy to Learn**
Simple syntax and clear documentation.



- **Large Community**
Huge developer community and strong ecosystem.



- **JSX Syntax**
JSX makes the code more readable and easier to write.

Key Takeaway :



React helps us build **modern, fast** and **scalable** web applications using **reusable components**.

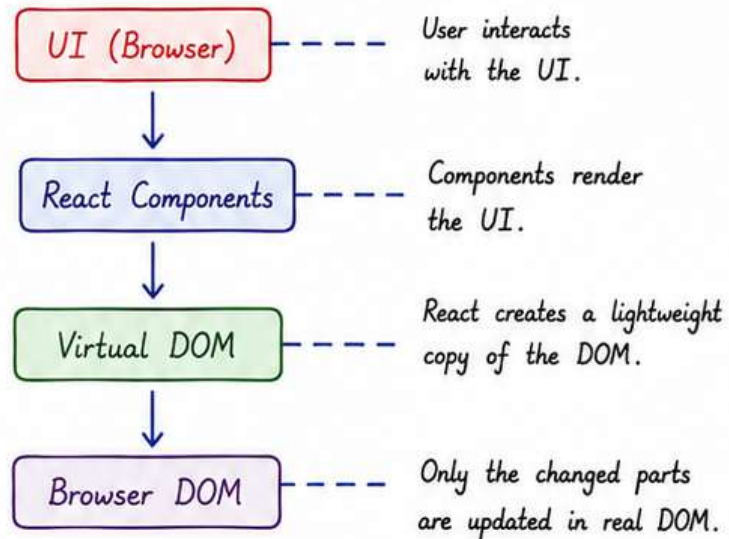


1.3 React Architecture

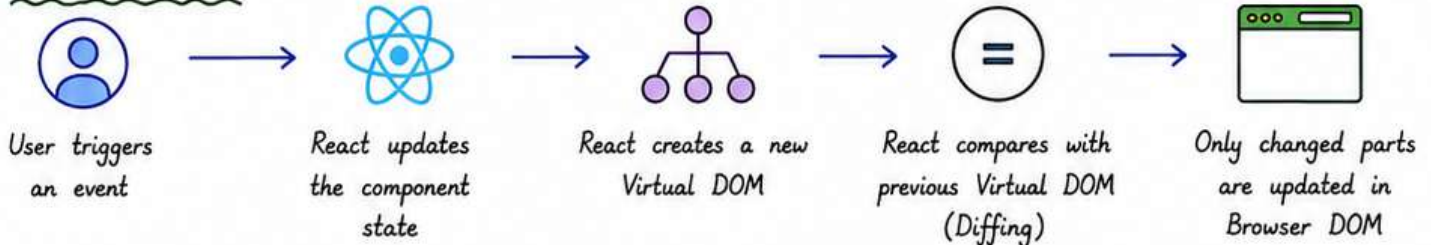
- * React follows a **component-based architecture**.
- * It uses **Virtual DOM** for efficient updates.
- * Data flows in **one direction** (top-down).
- * React separates UI into small, **reusable components**.

Key Points :

- ✓ Component-Based Structure.
- ✓ Virtual DOM for better performance.
- ✓ One-Way Data Binding.
- ✓ Reusable and Maintainable Code.



How it works?



1.4 React vs Traditional JavaScript

Feature	Traditional JavaScript	React (Modern JavaScript)
Approach	DOM Manipulation	Component-Based Approach
DOM Updates	Directly updates the real DOM (whenever data changes)	Uses Virtual DOM to update only changed parts
Code Structure	Hard to manage in large applications	Easy to manage with reusable components
Reusability	Code is not easily reusable	Components can be reused multiple times
Performance	Slower for large applications (due to direct DOM updates)	Faster and efficient (thanks to Virtual DOM)
Data Binding	Two-way binding (complex)	One-way binding (simple and predictable)
Maintainability	Hard to maintain and debug	Easy to maintain and debug
Learning Curve	Easier for beginners	Slightly higher, but worth learning



Key Takeaway :



React improves **performance**, **scalability** and **maintainability** by using **components** and **Virtual DOM**.



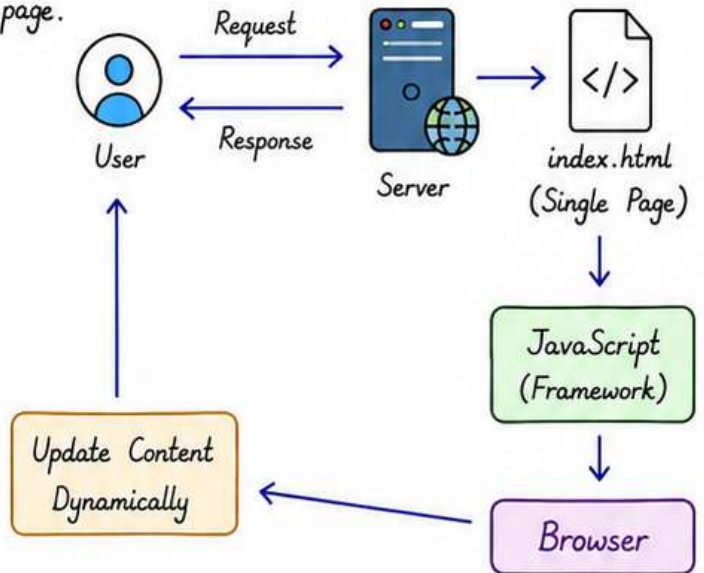
1.5 SPA Architecture

- * SPA stands for **Single Page Application**.
- * In SPA, the entire application runs on a **single page**.
- * Only a **single HTML page** is loaded from the server.
- * Content is updated **dynamically** without refreshing the page.
- * It provides a **smooth user experience** similar to desktop applications.

Key Points :

- ✓ Only one page (index.html) is loaded.
- ✓ Navigation happens without full page reload.
- ✓ Uses JavaScript to update content.
- ✓ Faster and better user experience.
- ✓ Examples : Gmail, Facebook, Twitter, etc.

How SPA Works?



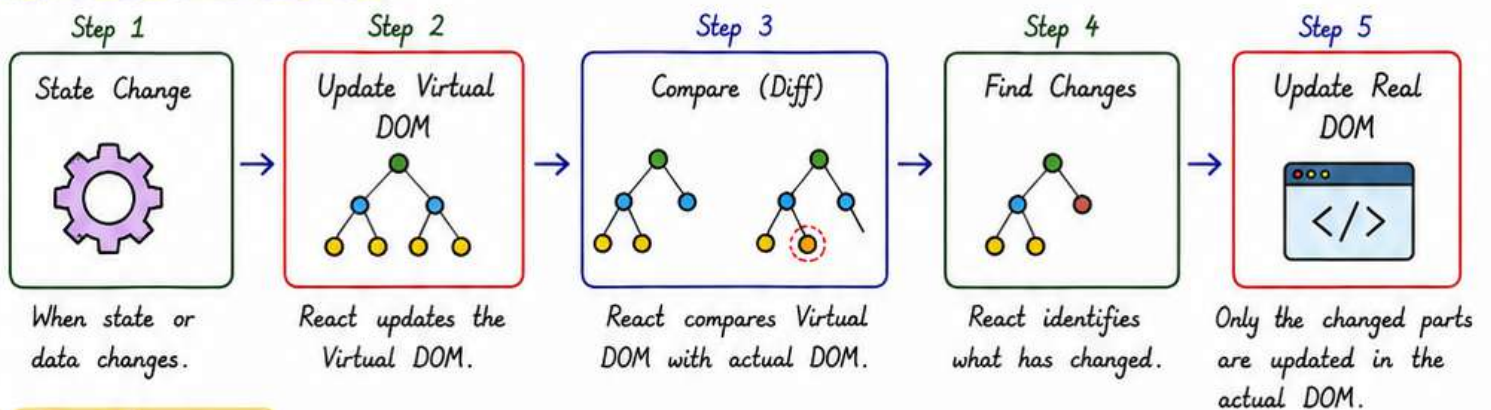
1.6 Virtual DOM

- * Virtual DOM is a **lightweight** copy of the actual DOM.
- * When data changes, React first **updates the Virtual DOM**.
- * It then **compares** the Virtual DOM with the actual DOM (**Diffing**).
- * Only the **changed parts** are updated in the actual DOM.
- * This makes React applications **faster** and **more efficient**.

Benefits of Virtual DOM

- ◆ Improves performance
- ◆ Reduces memory usage
- ◆ Updates only what is necessary
- ◆ Makes React very efficient

How Virtual DOM Works?



Key Takeaway :



Virtual DOM is the secret behind React's **high performance**. It updates only the **necessary parts** of the UI, not the entire page.



1.7 React DOM

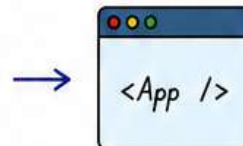
- ★ React DOM is the **entry point** to the **DOM** and server renderers for React.
- ★ It provides DOM-specific methods that can be used at the top level of your web app.
- ★ It is used to **render** React components into the actual DOM.
- ★ ReactDOM **creates** and **manages** DOM nodes efficiently.

Key Points :

- ✓ It acts as a bridge between React and the Browser DOM.
- ✓ It updates the DOM in the most efficient way.
- ✓ It provides methods like `render()`, `createRoot()`, `hydrateRoot()`, etc.
- ✓ It is a part of the `react-dom` package.

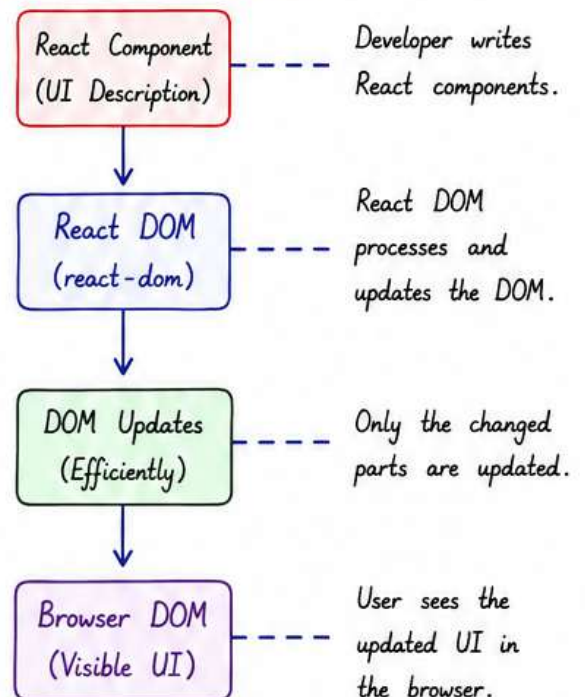
Example :

```
import ReactDOM from 'react-dom/client';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
```



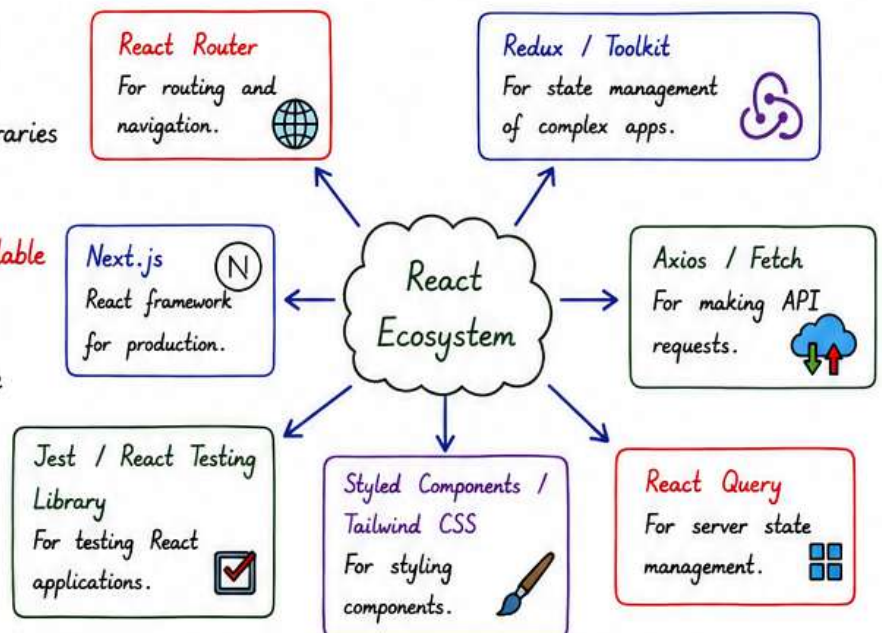
Renders the `<App />` component into the real DOM.

How React DOM Works?



1.8 React Ecosystem

- ★ React has a **rich ecosystem** of tools, libraries and frameworks.
- ★ These tools help in building complete, **scalable** and **maintainable** applications.
- ★ They solve common problems and improve **developer productivity**.



Key Takeaway :



React DOM helps React interact with the real DOM **efficiently**. The React ecosystem provides powerful tools to build **modern, scalable** and **high-performance** applications.



1.9 React Installation

- ★ Before installing React, we need **Node.js** and **npm** installed in our system.
- ★ There are two ways to set up a React development environment.
- ★ The most common way is using **Create React App (CRA)**.
- ★ It sets up a React project with the best configuration.

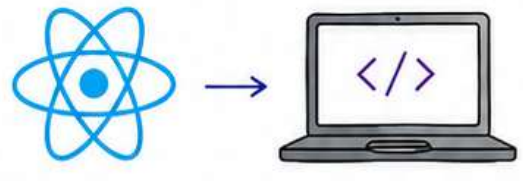
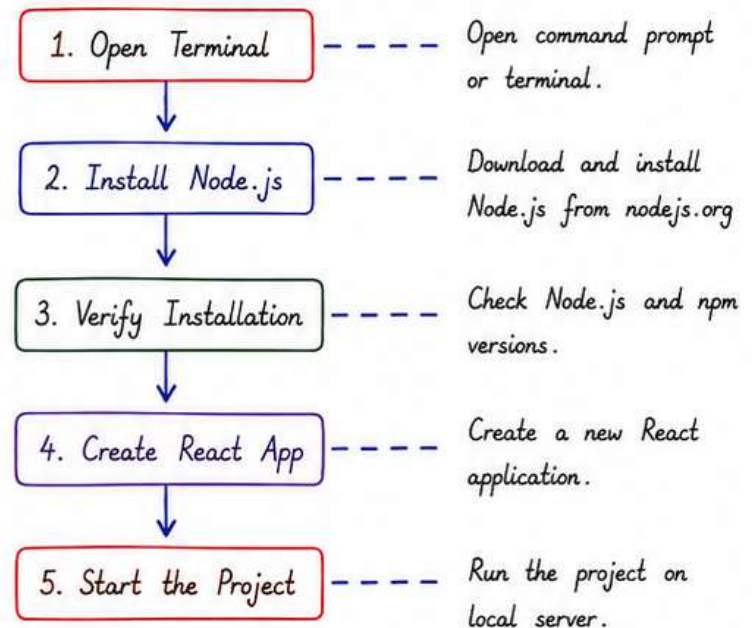
Key Points :

- ✓ Node.js and npm are required.
- ✓ Create React App is the easiest way.
- ✓ It creates a ready-to-use project structure.
- ✓ It includes best tools and configurations.

Commands

```
node -v           // Check Node.js version
npm -v            // Check npm version
npx create-react-app my-app // Create new React app
cd my-app         // Go to project folder
npm start         // Start development server
```

Steps to Install React (Using CRA)



After running these commands, your React app will run at <http://localhost:3000>



1.10 Node.js and npm

- ★ **Node.js** is a JavaScript runtime environment.
- ★ It allows JavaScript to run outside the browser.
- ★ **npm** (Node Package Manager) is used to install and manage packages.
- ★ React and many other tools are installed using npm.

Key Points :

- ✓ Node.js provides the environment.
- ✓ npm helps in package management.
- ✓ Together they power React development.
- ✓ Most React tools come from npm registry.

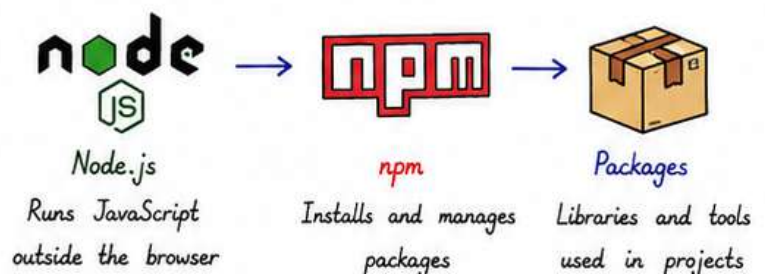
Key Takeaway :



React requires **Node.js** and **npm** to work.

They help us **install packages**, **run tools** and build **powerful applications**.

Node.js and npm Workflow



Common npm Commands

```
npm -v           // Check npm version
npm install <package> // Install a package
npm install       // Install all dependencies
npm uninstall <package> // Remove a package
```



1.11 Creating React Applications

- ★ The easiest way to create a new React application is by using **Create React App (CRA)**.
- ★ CRA sets up a modern React project with **zero** configuration.
- ★ It provides a ready-to-use project structure with best tools and configurations.
- ★ It is maintained by the **official React team**.

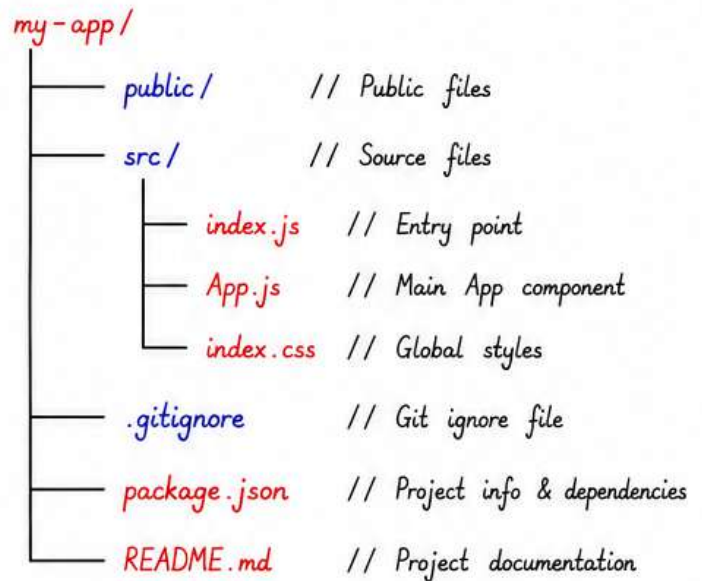
Key Points :

- ✓ Creates a new React project in seconds.
- ✓ No need to configure Webpack, Babel, ESLint, etc.
- ✓ Built-in support for development server.
- ✓ Best choice for beginners and quick start.

Commands

```
npm create-react-app my-app // Create a new React app
cd my-app // Go to project folder
npm start // Start development server
npm run build // Build for production
npm test // Run tests
npm eject // Eject configuration (not recommended)
```

Project Structure (CRA)



After running **npm start**

Your React application will run on
http://localhost:3000
in the browser.



1.12 Vite

- ★ Vite is a **next generation** build tool for modern web development.
- ★ It is extremely **fast** and **lightweight**.
- ★ Vite provides **instant** server start and **blazing fast** HMR (Hot Module Replacement).
- ★ It uses **ES Modules** and native browser support.

Key Points :

- ✓ Very fast development experience.
- ✓ Minimal configuration.
- ✓ Supports modern JavaScript features.
- ✓ Recommended for large and modern projects.

Vite vs CRA

Feature	CRA	Vite
Build Tool	Webpack	esbuild + Rollup
Startup Time	Slower	Very Fast
HMR	Slower	Instant
Configuration	Complex (Eject needed)	Simple
Bundle Size	Larger	Smaller
Performance	Good	Excellent

Commands (Vite)

```
npm create vite@latest my-vite-app -- --template react
// Create React app with Vite
cd my-vite-app // Go to project folder
npm install // Install dependencies
npm run dev // Start development server
npm run build // Build for production
npm run preview // Preview production build
```



Key Takeaway :



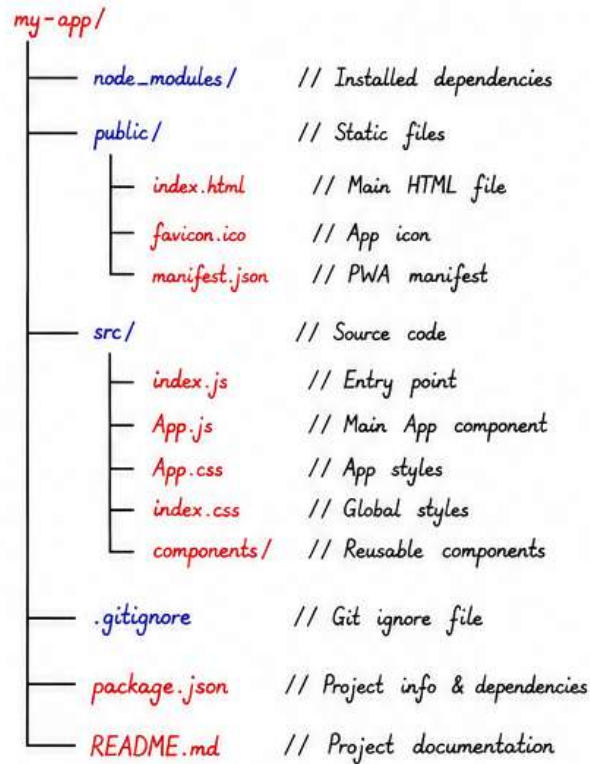
CRA is great for **beginners** and quick setup.
Vite is the future for **fast, modern** and **scalable** React development.



1.13 React Project Structure

- ☆ A React project created using CRA follows a standard folder structure.
- ☆ This structure helps in organizing files in a scalable way.
- ☆ All the important files are inside the `src/` folder.
- ☆ Public assets and configuration files are in the root directory.

All your code goes in `src/` and React renders the App inside `public/index.html`.



Key Points :

- ✓ `src/` contains the application source code.
- ✓ `public/` contains static files.
- ✓ `package.json` manages project dependencies and scripts.
- ✓ `node_modules/` is auto-generated.

1.14 package.json

- ☆ `package.json` is the heart of any React project.
- ☆ It stores project information, `dependencies` and `scripts`.
- ☆ It is used by `npm` to manage the project.

Example package.json

```
{
  "name": "my-app",
  "version": "1.0.0",
  "private": true,
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  }
}
```

Fields Description

Field	Description
name	Name of the project
version	Version of the project
private	Prevents accidental publishing
dependencies	Required packages for the app
scripts	Custom npm commands
react	React library
react-dom	React DOM for web
react-scripts	Scripts and configurations used by CRA

Key Takeaway :

`package.json` tells `npm` how to run the project and what to install.



1.15 npm Scripts

- ☆ `npm scripts` are predefined commands defined in `package.json`.
- ☆ They help to run, build, test and manage the React application easily.

Common npm Scripts in React

Script	Command	Description
start	npm start	Runs the app in development mode. Opens at <code>http://localhost:3000</code>
build	npm run build	Builds the app for production (Creates <code>build/</code> folder)
test	npm test	Runs tests in interactive watch mode.
eject	npm run eject	Ejects from CRA (One-way operation).

Instead of remembering long commands, we just use simple npm scripts.



Key Takeaway :



- ✓ npm scripts make development faster and easier.
- ✓ Most common scripts: `start`, `build`, `test`.

