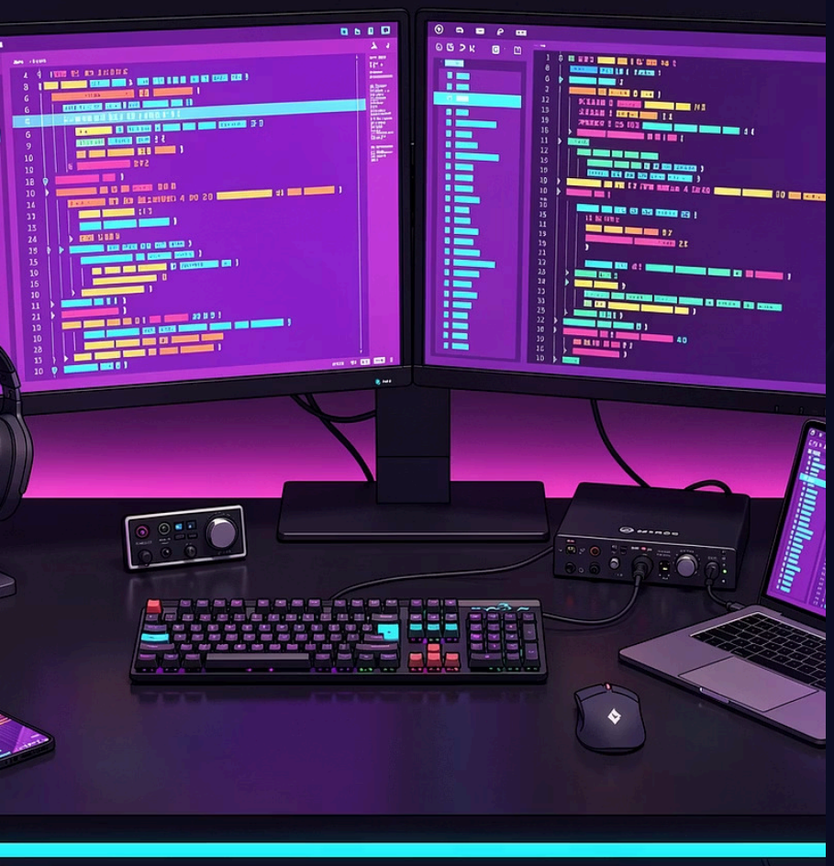


React JS Course

Build Modern Frontend Web Applications Using React JS — a comprehensive, hands-on journey from zero to deployment, designed for beginners and intermediate developers ready to level up their skills.

14 MODULES

BEGINNER TO INTERMEDIATE

HANDS-ON PROJECTS

Course Overview

This course is structured into 14 progressive modules — each one building on the last. From React fundamentals to advanced performance patterns, you'll gain the practical skills needed to build, style, and deploy real-world web applications.

01

Foundations

Modules 1–3: React basics, environment setup, JSX, components, props, and state

02

Core Skills

Modules 4–8: Component building, Hooks, rendering patterns, forms, and styling

03

Real-World React

Modules 9–12: Routing, state management, API integration, and advanced concepts

04

Build & Ship

Modules 13–14: Full project work and deploying live apps to Netlify, Vercel, and GitHub Pages

MODULE 1

Introduction to React

What Is React?

React is a declarative, component-based JavaScript library built and maintained by Meta (Facebook). It focuses on building reusable UI components and managing how your interface updates when data changes — making it one of the most popular frontend tools in the world.

How Does It Compare?

React vs. Vanilla JS

React abstracts DOM manipulation, making complex UIs far easier to manage

React vs. Angular

React is a library; Angular is a full framework — React gives you more flexibility

React vs. Vue

Both are component-based; React has a larger ecosystem and job market presence

React's Key Features & Advantages

React's design philosophy makes it uniquely powerful for building modern, scalable UIs. Here's what sets it apart — and where you should be aware of its limits.



Component-Based

Break UIs into small, self-contained components that are reusable across your entire application



Virtual DOM

React's Virtual DOM batches and minimizes real DOM updates, resulting in blazing-fast rendering



One-Way Data Flow

Data flows top-down through props, making your app's behavior predictable and easy to debug



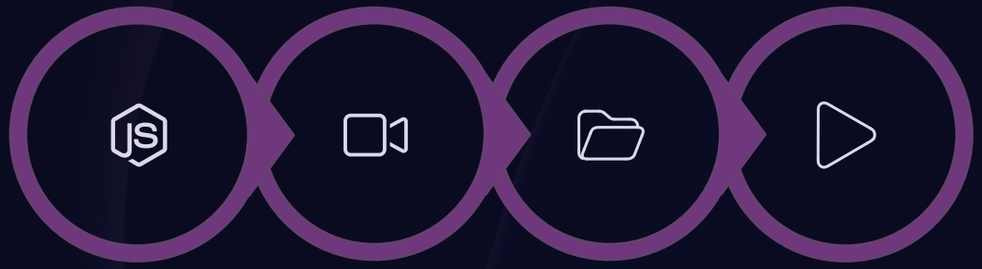
Limitations

React is a UI library, not a full framework — routing, state management, and HTTP need third-party tools

MODULE 2

Environment Setup

Before writing your first line of React, you need the right tools in place. This module walks you through installing Node.js and npm, scaffolding a project, and understanding the folder structure that React gives you out of the box.



Install Node.js | Create React App | Inspect Structure | Run Dev Server

Once your environment is running, you'll see your first React app live in the browser — a satisfying milestone that confirms everything is wired up correctly.



MODULE 3

React Basics

This is where React clicks. You'll learn the five building blocks that power every React application — master these, and everything else falls into place.

JSX

JavaScript XML — write HTML-like syntax directly inside JS. React compiles it to `React.createElement()` calls under the hood

Components

Functional components (modern, preferred) and class components (legacy but important to recognize)

Props

Pass read-only data from parent to child components, making your components dynamic and configurable

State

Local, mutable data that lives inside a component and triggers a re-render whenever it changes

Events

Handle user interactions like clicks, key presses, and form inputs using React's synthetic event system

MODULE 4

Component Building

Great React apps are built from small, focused, reusable components. This module teaches you how to design components that work together elegantly through nesting, composition, and thoughtful data flow.

Reusable Components

Design components that accept props and render differently based on inputs — write once, use everywhere

Nesting & Composition

Compose complex UIs by nesting smaller components inside larger ones — like building with Lego bricks

Parent & Child Communication

Pass data down via props and communicate up using callback functions — the backbone of React data flow



MODULE 5

React Hooks

Hooks revolutionized how we write React. Introduced in React 16.8, they let functional components manage state, side effects, and shared logic — no class components required.



useState

Declare and update local component state with a simple getter/setter pair



useEffect

Run side effects like fetching data, subscriptions, or DOM manipulation after renders



useContext

Consume values from React's Context API without prop drilling through every level



useRef

Access DOM elements directly or persist a mutable value across renders without triggering re-renders



useMemo & useCallback

Memoize expensive calculations and stable function references to prevent unnecessary re-renders



Custom Hooks

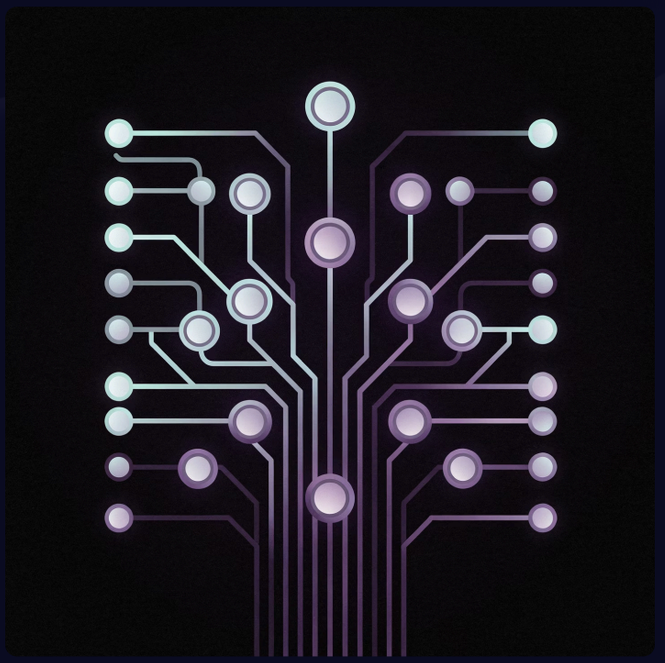
Extract and reuse stateful logic across components by composing built-in hooks into your own

MODULE 6

Conditional Rendering

Controlling What Gets Shown

React gives you multiple powerful patterns for rendering UI conditionally based on state, props, or any JavaScript expression. Picking the right pattern keeps your JSX clean and readable.



If / Else Rendering

Use standard JavaScript `if` statements outside JSX to decide which component or element to return

Ternary Operator

Inline `condition ? A : B` is perfect for toggling between two elements directly inside JSX

Logical && Short-Circuit

Render an element only when a condition is true — ideal for showing badges, alerts, or optional UI

Rendering Lists

Use `.map()` to transform arrays of data into lists of elements — always include a unique `key` prop

MODULE 7

Forms in React

Forms are at the heart of nearly every web app. React gives you two approaches to managing form data — controlled and uncontrolled components — plus powerful patterns for validation and submission.

Controlled Components

State drives the form input value. Every keystroke updates state via `onChange`, giving React full control — the recommended approach for most forms

Uncontrolled Components

The DOM manages the input value. Access it via a `useRef` reference — useful for simple forms or when integrating with non-React libraries

Validation & Submission

Validate inputs on change or on submit, display user-friendly error messages, and handle the submit event cleanly with `e.preventDefault()`

MODULE 8

Styling in React

React is styling-agnostic — you choose the approach that fits your project. This module covers five powerful methods, from quick inline styles all the way to utility-first frameworks.



Inline CSS

Apply styles directly as a JavaScript object — great for dynamic, computed styles



CSS Files & Modules

CSS Modules scope class names locally, preventing naming collisions across components



Bootstrap & Tailwind

Bootstrap offers ready-made components; Tailwind provides utility classes for rapid, custom styling



Styled Components

Write actual CSS inside JavaScript using tagged template literals — styles live alongside the component

MODULE 9

React Router

Single Page Applications

React apps are SPAs — the browser loads one HTML page and React swaps content dynamically. React Router manages this navigation without full page reloads, giving users a fast, app-like experience.

What You'll Learn

1. **Routes & Navigation**

   Define URL paths and map them to components using `<Route>` and `<Link>`

2. **Dynamic & Nested Routes**

   Use URL parameters (e.g., `/product/:id`) and nest routes for complex layouts

3. **Redirects & 404 Handling**

   Redirect users programmatically and gracefully handle unknown routes with a fallback page

MODULE 10

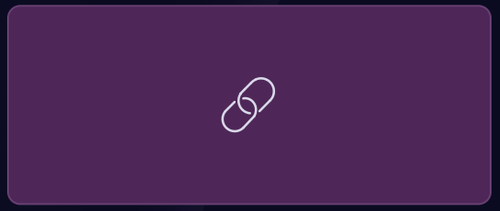
State Management

As apps grow, managing state across many components becomes challenging. This module takes you from simple state lifting all the way to Redux Toolkit — the industry standard for complex apps.



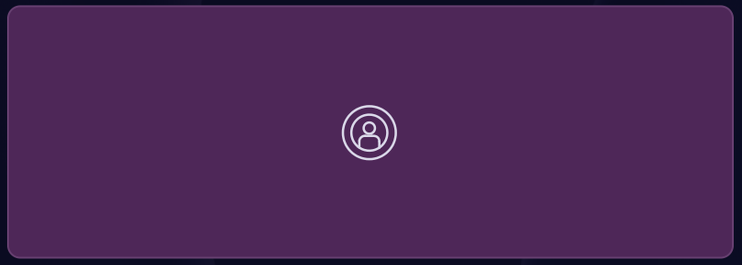
Lifting State Up

Move shared state to the nearest common ancestor



Prop Drilling

Understand the problem of passing props through many layers



Context API

Share state globally without prop drilling using React's built-in Context



Redux & Redux Toolkit

Centralize and manage application-wide state predictably with actions, reducers, and slices



MODULE 11

API Integration

Most real-world React apps talk to a backend. This module teaches you how to fetch, send, and display data from REST APIs — and handle loading and error states gracefully.

Fetch API & Axios

Use the browser's native `fetch` or the popular Axios library — both handle HTTP with promises and async/await

CRUD Requests

Perform GET (read), POST (create), PUT (update), and DELETE operations against any REST API

Loading & Error States

Show loading spinners during requests and display user-friendly error messages when things go wrong

Displaying API Data

Map over API response arrays to render dynamic lists, cards, and data-driven interfaces

MODULE 12

Advanced React Concepts

Ready to think like a senior React developer? This module dives into the internals and optimization techniques that separate good React apps from great ones.



Reconciliation & Virtual DOM

Understand how React compares the old and new Virtual DOM trees (diffing) to make surgical, efficient updates to the real DOM



Memoization & Optimization

Use `React.memo`, `useMemo`, and `useCallback` to skip unnecessary re-renders and boost performance



Lazy Loading & Code Splitting

Dynamically import components with `React.lazy` and `Suspense` to reduce initial bundle size and improve load times



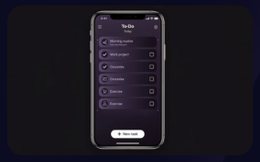
Error Boundaries

Wrap component trees with error boundaries to catch runtime errors gracefully and display fallback UIs instead of crashing the whole app

MODULE 13

React Project Work

The best way to solidify your React skills is by building real projects. This module takes you through 7 hands-on builds — ranging from beginner-friendly apps to full-featured applications you can add to your portfolio.



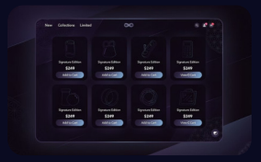
To-Do & Calculator

State management fundamentals — add, complete, and delete tasks; build arithmetic logic



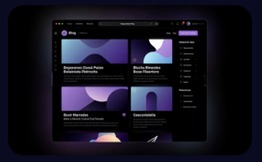
Weather & Quiz App

API integration with real weather data; interactive quiz with score tracking and conditional rendering



E-Commerce Product List

Filter and sort products fetched from an API, add to cart — realistic shopping UI patterns



Blog & CRUD App

Full CRUD operations — create, read, update, and delete content with React Router and API calls

MODULE 14

Deployment

Your app is built — now it's time to share it with the world. This final module walks you through creating a production-optimized build and deploying to three of the most popular free hosting platforms.

Production Build

Run `npm run build` to compile and optimize your React app into a static bundle — minified JavaScript, hashed filenames, and stripped dev warnings. This is what you deploy.



Netlify

Drag-and-drop your build folder or connect your GitHub repo for automatic deploys on every push



Vercel

Zero-config deployment with instant global CDN — the go-to choice for React and Next.js projects



GitHub Pages

Deploy static React apps directly from your GitHub repository for free — perfect for portfolios

You're Ready to Build 🚀

From setting up your first React app to deploying polished projects to the web — you now have everything you need to build modern, real-world frontend applications with confidence.

14 Modules

A complete, structured learning path from React basics to advanced patterns

7 Projects

Portfolio-ready apps that demonstrate your skills to future employers

1 Goal

Become a confident React developer who can build and ship production applications

