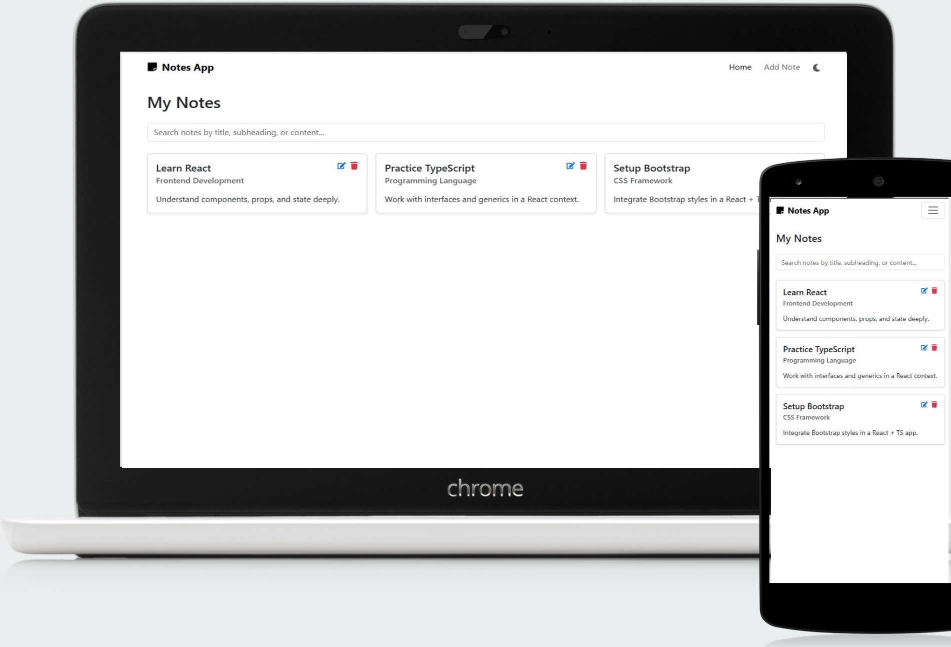
React Workshop

By TOP ENGINEERS

TOP ENGINEERS
UPGRADING ENGINEERING VERSION



Notes App

Home Add Note

My Notes

Search notes by title, subheading, or content...

Learn React
Frontend Development
Understand components, props, and state deeply.

Practice TypeScript
Programming Language
Work with interfaces and generics in a React context.

Setup Bootstrap
CSS Framework
Integrate Bootstrap styles in a React + T

Notes App

My Notes

Search notes by title, subheading, or content...

Learn React
Frontend Development
Understand components, props, and state deeply.

Practice TypeScript
Programming Language
Work with interfaces and generics in a React context.

Setup Bootstrap
CSS Framework
Integrate Bootstrap styles in a React + TS app.

Prerequisite

Install VS Code	https://code.visualstudio.com/download
Install Git	https://git-scm.com/install
Vercel	https://vercel.com
Git Hub	https://github.com
Install Node	https://nodejs.org/en/download

Why Frontend Development? And Where it stands:

- Frontend is what users directly interact with
- It focuses on user experience, usability, and design
- Connects visual interface to backend systems
- Defines how users perceive and navigate digital products
- Sits between UI design and backend logic.
- Acts as a bridge converting business logic into user actions

React, Angular, and Vue

- Each solves frontend challenges differently
- Angular and Vue are frameworks: full ecosystem for routing, state, forms, etc
- React is a library: focused only on UI, gives freedom to pick other tools
- Angular = opinionated and structured
- React = flexible and modular
- Vue = balanced between simplicity and structure

JSX and How React Handles It

- JSX = JavaScript XML: allows writing HTML in JavaScript
- Transpiled by Babel into React.createElement() calls
- Enables declarative UI building

Example:

const element = <h1>Hello, world!</h1>

React converts this into efficient virtual DOM elements

State

- Passed from parent to child components
- Simiilar to Function Args
- Immutable - Read-only
- Used for component configuration

Props

- Managed within the component
- Similar to local variables
- Mutable - changes trigger re-render
- Used to track dynamic data or UI updates

React Hooks

Introduced in React 16.8 to use state and lifecycle features in functional components

- **useState**: Manage local state
- **useEffect**: Handle side effects (like fetching data)
- **useContext**: Access global context
- **useRef**: Reference DOM elements
- **useMemo** & **useCallback**: Optimize performance
- **Custom Hooks**: Created by combining existing hooks for reusable logic

Bootstrap

- CSS framework for responsive design
- Uses grid system (rows, columns)
- Predefined utility classes for margin, padding, typography
- Makes UI development faster and consistent

Example classes

- ➔ *.container, .row, .col*
- ➔ *.btn, .btn-primary*
- ➔ *.text-center, .mt-3*

React Context and Providers

- Context helps pass data across components without prop drilling
- Provider makes data available to its descendants
- Consumer (or useContext) reads the data

Example

const MyContext = React.createContext();

<MyContext.Provider value={data}>...</MyContext.Provider>

Web Communication & REST APIs

- Frontend communicates with backend via APIs
- REST = Representational State Transfer
- Stateless communication using HTTP methods
- Frontend sends requests; backend responds with JSON/XML
- Enables data fetching, posting, updating, and deletion

HTTP Methods (REST API)

- GET → Retrieve data
- POST → Create new resource
- PUT → Update existing resource
- PATCH → Partially update resource
- DELETE → Remove resource

Questions?

Thank You

TOP ENGINEERS

UPGRADING ENGINEERING VERSION