Forsk Coding School

Powered Learning for Future Success

COURSE BROCHURE / JAIPUR

FRONTEND

Development with React.js



Create responsive websites.
Build interactive React applications.
Manage state with Redux Toolkit.

FROM YOUR FIRST PAGE TO A WORKING APP

Design the interface. Bring it to life.

01

TRAIN

Web Design, JavaScript, React, Redux & RTK

02

INTERN

Full-time internship at Forsk Technologies

03

INTERVIEW

Preparation, feedback and placement support

FULL-TIME INTERNSHIP WITH

Forsk Technologies
From AI to AGI – We Build What's Next

YOUR FRONTEND LEARNING PATH

Start with the web.
Progress into React.

Learn the foundations, add behaviour with JavaScript and build reusable, data-driven interfaces with React and Redux Toolkit.

WEB FOUNDATIONS

01
HTML & CSS
Structure and styling

02
Responsive layouts
Flexbox, Grid and forms

03
Bootstrap & Tailwind
Practical UI building blocks

JAVASCRIPT

04
Core JavaScript
Logic, DOM and events

05
ES6 to ES10
Modern syntax and modules

06
Async & APIs
Fetch, promises and errors

REACT APPLICATIONS

07
React & hooks
Components, props and state

08
Routing & forms
Pages and user interaction

09
Redux & RTK
Shared state and data flows

WHO CAN START HERE

Beginners, college students, graduates and career switchers who want to build practical frontend skills through consistent coding practice.

YOUR PORTFOLIO DIRECTION

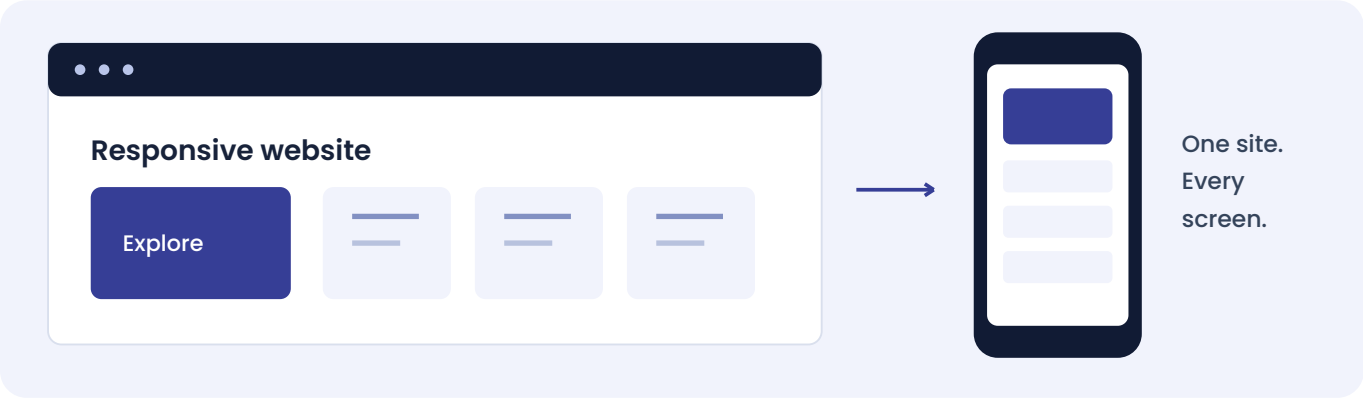
Build responsive interfaces you can explain.

Work towards Frontend Developer, React Developer and Web UI Developer roles.

MODULE 01 / WEB DESIGN

Build the structure.
Design for every screen.

Learn HTML, CSS, Bootstrap, Tailwind CSS and Core JavaScript. Turn a page layout into a responsive, interactive website.



HTML foundations

Semantic pages, links, images, lists and forms.

CSS layouts

Selectors, the box model, Flexbox and Grid.

Bootstrap

Responsive grids, components and utilities.

Tailwind CSS

Utility classes, spacing and responsive styling.

Core JavaScript

Variables, conditions, loops, functions and DOM events.

Usable interfaces

Readable layouts, keyboard access and form feedback.

PRACTICE PROJECT

A responsive business website or portfolio

Build navigation, content sections and an interactive enquiry-form interface.

MODULE 01 / LAYOUT & USABILITY

Make layouts adapt.
Keep every control usable.

Go beyond a single screen. Practise page structure, flexible layouts and clear interactions, then apply Bootstrap and Tailwind CSS.

01 **Structure with meaning**

Use headings, landmarks, links, images and labelled form fields. Choose elements that match the content and interaction.

02 **Control the CSS**

Work with selectors, inheritance, specificity, spacing, typography and the box model. Inspect styles in browser developer tools.

03 **Build flexible layouts**

Combine Flexbox, Grid, relative sizing and media queries. Check content wrapping, navigation and images at different viewport widths.

04 **Make the page usable**

Practise keyboard navigation, visible focus, useful image descriptions and clear form messages. Check readable text and colour contrast.

TWO STYLING APPROACHES

Bootstrap

Practise its grid, spacing utilities, forms and reusable components. Customise a consistent page layout.

Tailwind CSS

Compose interfaces with utility classes. Practise responsive variants, interactive states and consistent design values.

LAYOUT LAB

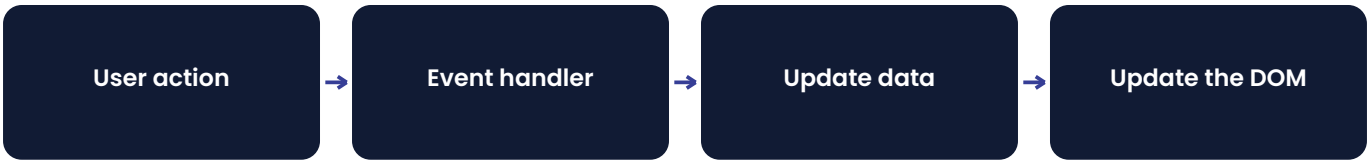
Rebuild one layout in two approaches.

Create a responsive landing page, then compare its Bootstrap and Tailwind implementations.

MODULE 01 / CORE JAVASCRIPT

Turn static pages into working interfaces.

Build a programming foundation before moving into React. Connect values, conditions and functions to real browser interactions.



Values & decisions

Variables, strings, numbers, booleans, operators and comparisons. Use conditions to decide what an interface should do.

Loops & functions

Repeat work with loops, create functions with parameters and return values, and understand scope and reusable logic.

Arrays & objects

Model lists and records. Read, add, update and remove values; practise transforming data for a page.

DOM & events

Select elements, update content and classes, and handle clicks, input and form submission through event listeners.

Forms & browser storage

Validate input, display helpful messages and practise localStorage for non-sensitive preferences. Work with JSON data.

Debugging habits

Use console output, breakpoints and the call stack. Reproduce an error, inspect values and explain the correction.

JAVASCRIPT LAB

Build a task list with filters and saved preferences.

Create, complete and remove tasks; validate input and explain each event handler.

MODULE 02 / ADVANCED JAVASCRIPT

Add behaviour.
Work confidently with data.

Build on Core JavaScript with ES6 to ES10, reusable modules, asynchronous code and API-driven interfaces.

01 **Modern JavaScript**

Let and const, arrow functions, template literals, destructuring, spread syntax and modules.

02 **Arrays, objects & logic**

Transform collections, work with objects and JSON, and organise reusable functions.

03 **Asynchronous programming**

Promises, async/await and Fetch for requesting and processing application data.

04 **Reliable user experiences**

DOM interactions, error handling, loading and empty states, browser tools and debugging.

MAKE EVERY STATE CLEAR

IDLE

Waiting for user input

LOADING

Request is in progress

SUCCESS

Show the returned data

ERROR

Explain what went wrong

PRACTICE PROJECT

An API-powered search interface

Fetch results, filter or sort data and handle loading, empty and error states.

MODULE 02 / API INTEGRATION

Request the data.
Handle every outcome.

Connect a frontend to an existing or practice API. Understand the request, inspect the response and keep the interface useful when conditions change.

01 / REQUESTS

HTTP & JSON

Practise URLs, query parameters, methods, headers and JSON payloads. Inspect requests and response data in the browser network panel.

02 / ASYNC FLOW

Promises & async/await

Follow asynchronous execution, consume Fetch responses and handle failures. Check HTTP status as well as network errors.

03 / RESPONSE HANDLING

Use the result carefully

Parse JSON, check the expected data shape and render results. Practise cancelling outdated requests when user input changes.

04 / USER EXPERIENCE

Keep feedback clear

Show loading, empty, success and error states. Offer a useful retry path and avoid displaying stale results as current data.

PRACTISE CHANGING CONDITIONS

Try a slow response, no matches, a failed request and repeated searches.

API LAB

Build a searchable catalogue from a practice API.

Add filters and pagination where the API supports them; show clear loading and error feedback.

MODULE 03 / REACT.JS

Think in components. Build interactive apps.

Break an interface into reusable components, pass data through props and respond to user actions with state and hooks.

01 / COMPOSITION

Components & JSX

Create reusable components, pass props and render conditions and lists with suitable keys.

02 / INTERACTION

State & hooks

Handle events and practise useState, useEffect, refs and reusable custom hooks.

03 / USER INPUT

Forms & validation

Manage controlled inputs, validate values and give clear feedback during form submission.

04 / APPLICATIONS

Routing & API integration

Organise page navigation, request data and handle loading, empty and error states.

THE REACT INTERACTION LOOP

User event → State update → Render → Updated UI

Practise a searchable product listing or task board using reusable React components.

MODULE 03 / REACT IN PRACTICE

Choose the right state.
Keep components clear.

Use hooks with a purpose. Separate user interactions, derived values and external synchronisation so application behaviour stays understandable.

State updates

Practise useState with objects and arrays. Use an updater function when a new value depends on the previous value.

Derived values

Calculate filtered lists and totals from props or state. Avoid storing duplicate values that can fall out of sync.

Effects & cleanup

Use useEffect for external synchronisation. Understand dependencies and clean up listeners, timers or requests when appropriate.

Refs & focus

Use useRef for DOM access and values that do not drive rendering. Practise focusing an input after an interaction.

Custom hooks

Extract reusable behaviour into custom hooks. Give the hook clear inputs and outputs and keep components focused on their interface.

Where state belongs

Keep temporary state close to its use. Lift shared state to a common parent and practise Context for wider access.

REACT LAB

Build a filterable dashboard with reusable hooks.

Explain which values are stored, which are derived and which interactions need an effect.

MODULE 03 / COMPLETE USER JOURNEYS

Make inputs reliable.
Make navigation clear.

Connect screens and interactions with React Router and form handling. Practise flows that still make sense when input is invalid or a URL is unknown.

FORMS & VALIDATION

Capture input

Controlled text fields, selects and checkboxes; field values and change handlers.

Validate clearly

Required values, formats and helpful field messages. Preserve useful input when a submission fails.

Handle submission

Pending, success and failure feedback. Prevent accidental repeated submissions and manage focus.

ROUTING & PAGE FLOWS

Map the screens

Create routes and shared layouts. Connect pages with links and meaningful navigation.

Use URL information

Read route parameters and query strings for detail pages, search terms and filters.

Handle the edges

Add a not-found view. Check direct links, refresh, back navigation and nested page layouts.

EXAMPLE ROUTES

/products /products/:productId Not-found page

USER-JOURNEY LAB

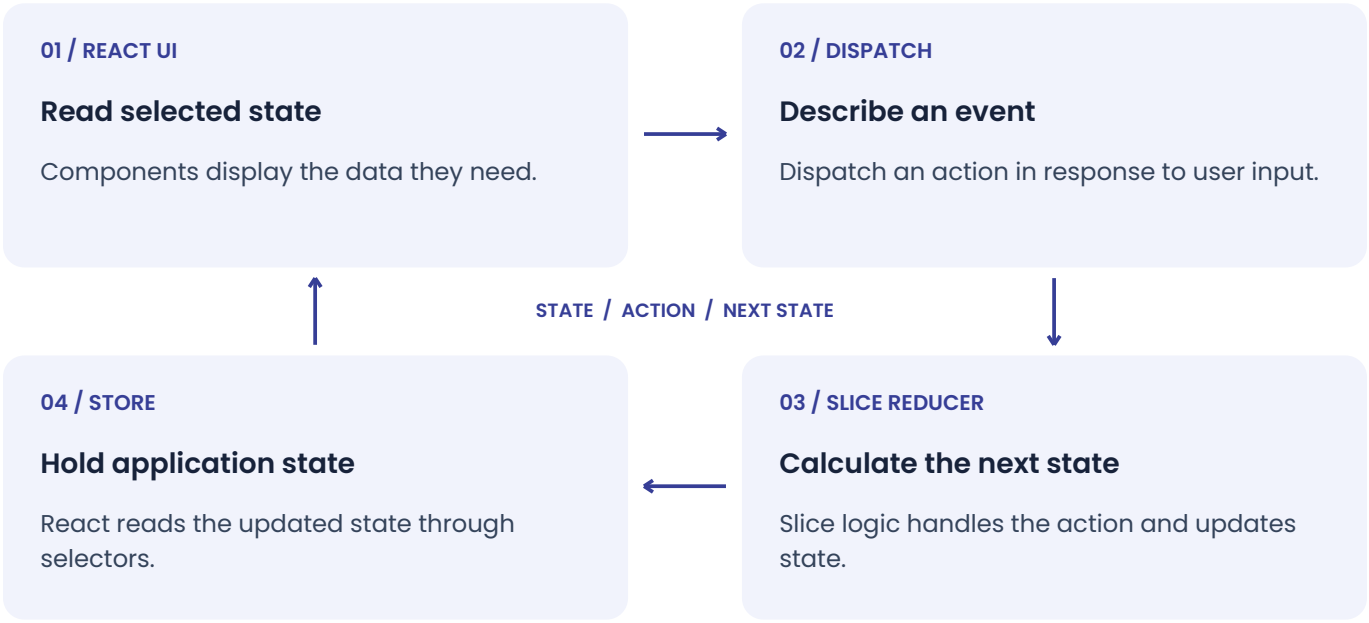
Build a catalogue, detail page and enquiry form.

Review validation, route changes and failure states using a practice API or mock data.

MODULE 03 / REDUX + RTK

Make shared state predictable.

Use Redux and Redux Toolkit (RTK) to organise shared application state and connect state changes to React interfaces.



Store & slices

configureStore, createSlice and action creators.

React-Redux hooks

useSelector, useDispatch and derived UI state.

Async workflows

Async thunks and pending, success and error states.

API state with RTK Query

Query and mutation basics, caching and refetching.

PRACTICE PROJECT

A shared cart or task-management feature

Keep items, quantities, filters and totals consistent across multiple screens.

MODULE 03 / APPLICATION DATA

Know where each piece of data belongs.

Distinguish local interaction state, shared client state and data from a server. Choose a clear approach for each part of the application.

LOCAL UI

An open dialog or an unfinished input usually belongs near the component that uses it.

SHARED CLIENT STATE

A cart or shared selection can live in a Redux slice and be read through selectors.

REMOTE DATA

An API catalogue needs fetching, loading feedback and a strategy for keeping cached results current.

RTK Query

Define API queries and mutations; use generated hooks to read request state. Practise caching, refetching and refreshing affected data after a change.

Async workflows

Practise createAsyncThunk when a workflow needs custom async logic. Handle pending, fulfilled and rejected outcomes in the slice and UI.

INSPECT THE DATA FLOW

Use Redux DevTools to inspect actions and state. Keep business data serialisable and calculate derived values with selectors.

DATA LAB

Connect a shared cart to an API-backed catalogue.

Show how local UI, cart state, remote data and request failures are handled separately.

PORTFOLIO PRACTICE

Build something you can demonstrate.

Choose a frontend project scenario, build the user journeys and show how the interface behaves with real application states.

01 Developer portfolio

Responsive pages, project showcases and contact UI.

02 React storefront

Product search, filters, cart state and checkout UI.

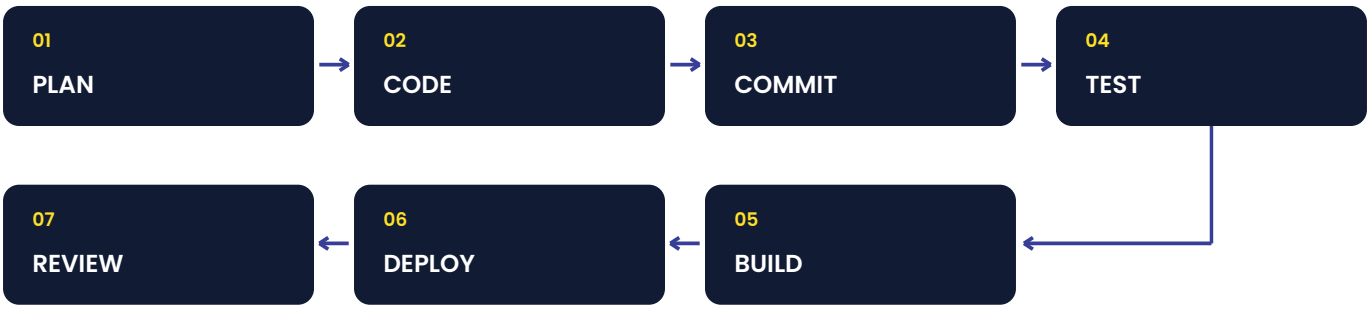
03 Task-management board

Add tasks, update status and manage shared filters.

04 Admin dashboard

API data, table views, forms and reusable components.

FROM A FEATURE TO A DEPLOYED FRONTEND



Review responsiveness, keyboard access, forms, API behaviour and browser errors.

CAPSTONE EVIDENCE

Make your work easy to review.

Include a Git repository, setup guide, screenshots and a working frontend demo.

CAPSTONE / BUILD, CHECK & EXPLAIN

Make the project work.
Make it easy to review.

Turn a chosen project into a complete frontend demo. Use these review areas to identify gaps and prepare to discuss your implementation.

REVIEW AREA	EVIDENCE TO DEMONSTRATE
Responsive UI	Check small and large screens, content wrapping, image sizing and navigation without horizontal overflow.
User interactions	Demonstrate form validation, search, filters, cart or task actions, and clear feedback after an action.
State & API logic	Explain state ownership, data transformations and API requests. Demonstrate empty, loading and error cases.
Accessibility	Check keyboard access, visible focus, meaningful labels and readable contrast in the main user journeys.
Code & Git	Organise components and modules, make useful commits and document setup, features and known limitations.
Build & deployment	Run a production build, configure public settings and check live routes, asset loading and browser errors.

PROJECT HANDOFF

A demo, a repository and a clear explanation.

Present the problem, key decisions, test cases, limitations and the improvements you would make next.

MODULE FEES / INDIAN RUPEES

One frontend pathway.
Three clear modules.

The complete Frontend with React.js pathway combines Web Design, Advanced JavaScript, React, Redux and Redux Toolkit.

COURSE MODULE	FEE (INR)
01 **Web Design** HTML, CSS (Bootstrap & Tailwind CSS), Core JavaScript	₹7,500
02 **Advanced JavaScript** ES6 to ES10	₹4,000
03 **React + Redux + RTK** Redux Toolkit (RTK)	₹8,500

TOTAL FRONTEND COURSE FEE

Web Design + Advanced JS + React / Redux / RTK

₹20,000

BUILD A STRONG FRONTEND FOUNDATION

Responsive layouts • JavaScript and APIs • React applications
Redux state management • Portfolio project practice

Complimentary database training applies to Full Stack course enrolments.

TRAINING TO FULL-TIME INTERNSHIP

Learn. Build.
Then put it into practice.

Your journey begins with training at Forsk Coding School and continues with a full-time internship at Forsk Technologies.

01 / TRAINING INSTITUTE

Forsk Coding School
Powered Learning for Future Success

Learn Frontend with React through structured course content, practical assignments and project work. Build the technical foundation you will carry into your internship.

↓ AFTER TRAINING

02 / FULL-TIME INTERNSHIP

Forsk Technologies
From AI to AGI – We Build What's Next

Apply your learning during a full-time internship at Forsk Technologies. Develop practical experience and prepare to explain your skills and project work in company interviews.

WHAT COMES NEXT

Company preparation and interviews.

Assessment, feedback and focused preparation continue across interview opportunities.

CAREER PREPARATION

Prepare. Interview. Improve. Repeat.

Company interviews may be remote or on site. Each round informs the preparation for the next opportunity.

01 / PREPARE

Company-specific preparation

Revise skills, practise questions and explain your project work.

02 / INTERVIEW

Meet the company

Attend the company's interview, remotely or on site.

FEEDBACK INTO PREPARATION

03 / ASSESS

Review the interview

Assess performance, discuss feedback and identify skill gaps.

04 / PREPARE AGAIN

Work on the gaps

Practise the weak areas and prepare for the next company.

When selected: progress through the company's offer and joining process. Selection depends on the employer's requirements and interview outcome.

SUPPORT THROUGH THE PROCESS

Resume & GitHub portfolio • Mock interviews • Project presentations

START YOUR FRONTEND JOURNEY

7231968183

forskcodingschool.com

COURSE DETAILS