# Introduction to React

Chitram Dasgupta

July 10, 2025

## Contents

# 1 What is React?

- React is a JavaScript library for building user interfaces.
- React makes it easier to build complex UIs using **components**.

# 2 Components in React

- We write React code using components.
- A component is a piece piece of UI that has its own logic and appearance.
- Think of a jigsaw puzzle: the puzzle is the react code, and each puzzle piece is a component.

## 2.1 Writing Components

- A component is a JavaScript function that returns **JSX**.
- Component names must start with an uppercase letter.

## 2.2 JSX

- JSX is a syntax extension of JavaScript that lets you write HTML-like markup inside a JavaScript file. We do not use plain HTML because HTML is static and cannot support conditionals, loops, variables, etc.
- Some differences with HTML are: `class` in HTML becomes `className` in JSX. `<label for="email">` becomes `<label htmlFor="email">`.
- We can evaluate JS expressions inside `{{ }}` in JSX.

## 2.3 Props

- Parent components can pass data to its child components using *props*.
- Remember that data flow is **one-way**, from **parent to child**, through **props**.

## 2.4 Example Component

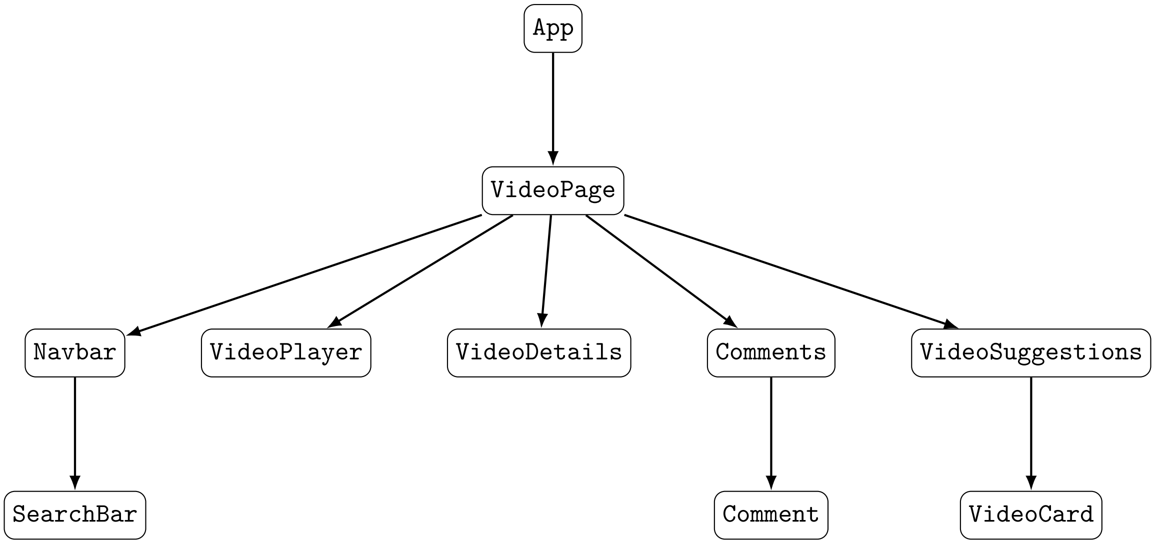
```
export default function ProfileDetails(props) {
  const { id, name, branch } = props.user;

  return (
    <article>
      <h2>Profile Details</h2>
      <p>ID: {id}</p>
      <p>Name: {name}</p>
      <p>Branch: {branch}</p>
    </article>
  );
}

<ProfileDetails
  user={{ id: 10, name: "Chitram Dasgupta",
          branch: "Computer Science & Engineering" }} />
```

# 3 Component Tree

Let's take a familiar example: simplified Youtube video page.



The components are:

1. `<App />`
2. `<VideoPage />`
3. `<Navbar />`
4. `<SearchBar />`
5. `<VideoPlayer video={selectedVideo} />`
6. `<VideoDetails video={selectedVideo} />`
7. `<Comments comments={selectedVideo.comments} />`
8. `<Comment comment={...} />`
9. `<VideoSuggestions videos={relatedVideos} onSelect={setSelectedVideo} />`
10. `<VideoCard video={...} />`

# 4 Rendering Strategies

The JSX returned by a component has to be processed *somewhere*. Where it is processed (that is, rendered) determines the rendering strategy.

Remember that, no matter what rendering strategy we use, the server will always respond with content. Rendering strategy determines the type of content returned by the server.

## 4.1 Client-side Rendering

On the first request, the server sends back an HTML file and a JS bundle.

The web browser downloads the content and renders the web page. React then takes over and renders subsequent content dynamically.

This naturally gives rise to **Single Page Applications**.

React historically supported only client-side rendering.

## 4.2 Server-side Rendering

The server pre-renders the web content in the server and then sends back the *rendered* content to the web browser.

**Important:** This rendering happens every time the client requests for this page, unless some caching mechanism is used.

We can do server-side rendering when React is used with a framework like Next.js.

## 4.3 Static Site Generation

The entire web application is compiled and then served directly.

Every time the client requests for web content, the server *does not render* the content again. It simply serves the rendered web content back to the client.

Frameworks like Next.js can achieve this.

# 5 Create a React Project

- Make sure you have **Node** installed (preferably, the latest *LTS* release).
- Run the following commands:

  ```
  npm create vite@latest react-bootcamp -- --template react
  ```

  ```
  cd react-bootcamp/
  ```

  ```
  npm install
  ```

  This will create a new React project.
- You can run the development server using:

  ```
  npm run dev
  ```
- Then open your web browser and paste in the URL: `http://localhost:5173/`