Introduction to React

🌀 What is React?

React is a **JavaScript library** (not a full-blown framework like Angular) made by Meta (Facebook). It helps you build **fast, interactive UIs** for web apps — especially SPAs (Single Page Applications).

Why React?

- Reusable components 🔁
- Super fast with virtual DOM ⚡
- Big ecosystem & community 🌐
- Easy to learn, but hella powerful 💪

⚔️ SPA vs MPA

SPA (Single Page Application):

- Loads a **single HTML file**, and dynamically updates content via JS.
- Feels super fast (no full reloads).
- React is BUILT for this.

🟢 Pros: Fast, fluid, app-like.

🔴 Cons: SEO can be tricky, initial load might be heavier.

MPA (Multi Page Application):

- Traditional style (like older websites).
- Every click loads a **new HTML page from server**.

🟢 Pros: Better SEO, simpler routing.

🔴 Cons: Slower, more page reloads.

👉 React = SPA king.

🛠️ Installation (CRA vs Vite)

📦 1. Create React App (CRA)

The OG, officially supported.

```
npx create-react-app my-app
cd my-app
npm start
```

🟢 Easy setup, works out the box.

🔴 Slower dev build time, chunky config.

⚡ 2. Vite (new hotness)

Vite = "vite" as in **speed** in French — super fast build tool.

```
npm create vite@latest my-app -- --template react
cd my-app
npm install
npm run dev
```

🟢 FAST af dev server, modern setup.

🔴 Slightly more setup to configure extras.

✅ I 100% recommend **Vite** for that Gen Z dev speed 🚀

📁 Folder Structure (default Vite/CRA)

Typical React app:

```
my-app/
├── public/        # Static files (index.html lives here)
├── src/           # All React code lives here
│   ├── App.jsx      # Main component
│   ├── main.jsx     # Entry point (where ReactDOM renders App)
│   ├── components/  # Custom components you make
│   └── assets/      # Images, icons, etc
├── package.json     # Dependencies & scripts
```

You'll mostly live inside src/ 💻

🧪 JSX & Babel

JSX = JavaScript + XML

Looks like HTML inside JS. React uses JSX to describe UI.

```
const App = () => {
  return <h1>Hello, Bruce 🦇!</h1>
}
```

Looks illegal but it's not 😅

Babel = the translator

JSX isn't real JS — Babel compiles it down to regular JS React code.

```
<h1>Hello</h1>
↓
React.createElement("h1", null, "Hello")
```

⚖️ Functional vs Class Components

Functional Component (modern way)

```
function Welcome(props) {
  return <h1>Hello {props.name}</h1>;
}
```

or even cleaner with arrow fn:

```
const Welcome = ({ name }) => <h1>Hello {name}</h1>;
```

✅ This is the **GOATED** way now — with Hooks, functional components can do everything.

Class Component (old school)

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello {this.props.name}</h1>;
  }
```

```
}
```

Not used much anymore unless you're working with legacy code.

📦 Props and State

🔗 Props (short for "properties")

- Passed **from parent to child**
- **Read-only**

```
const Greeting = ({ name }) => <h1>Hello {name}</h1>;

// usage
<Greeting name="Bruce" />
```

🧠 State

- **Internal data** of a component
- Can **change over time** (like UI reacting to user input)

With Hooks:

```
import { useState } from "react";

const Counter = () => {
  const [count, setCount] = useState(0); // state variable

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Add</button>
    </div>
  );
};
```

🧠 useState is like giving your component a memory.

Move to next, coder 👨‍💻💥