Step-by-Step Learning Plan of React Js

□ Part 1: Prerequisites

Before diving in, make sure you understand:

Topic	What to Know
HTML/CSS	Page structure and styling
JavaScript (ES6+)	Arrow functions, destructuring, promises
Basic Node.js	`npm`, `require`, modules

□ Part 2: Learn React.js (Frontend)

□ Step 1: Setup React Project

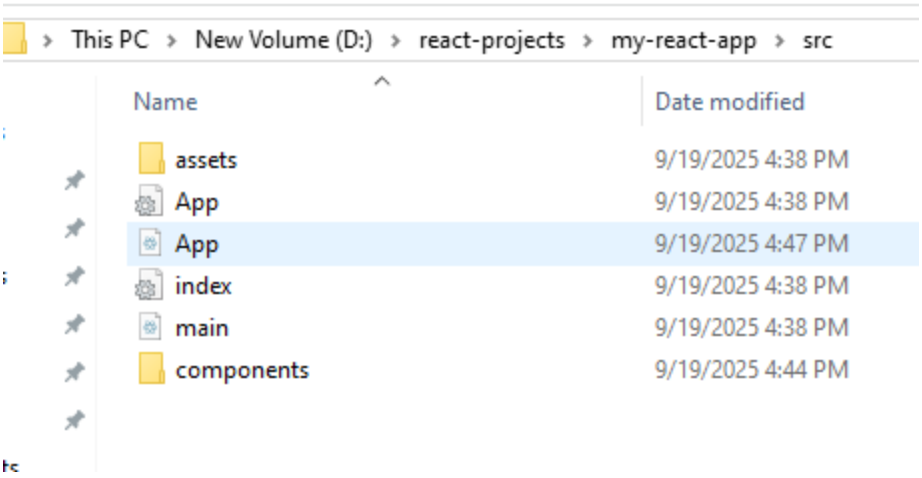
Use **Vite** (faster than CRA):

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

□ Folder structure:

```
my-react-app/
├─ src/
│  ├─ App.jsx
│  └─ main.jsx
```

As shown below structure:-



□ Step 2: Create a Component(`Greeting.jsx`):-

```
// src/components/Greeting.jsx
function Greeting({ name }) {
  return <h1>Hello, {name}!</h1>;
}

export default Greeting;
```

Use it in `App.jsx`:

```
// src/App.jsx

import Greeting from './components/Greeting';


function App() {

  return (

    <div>

      <Greeting name="Hello Om sir" />

    </div>

  );

}
```

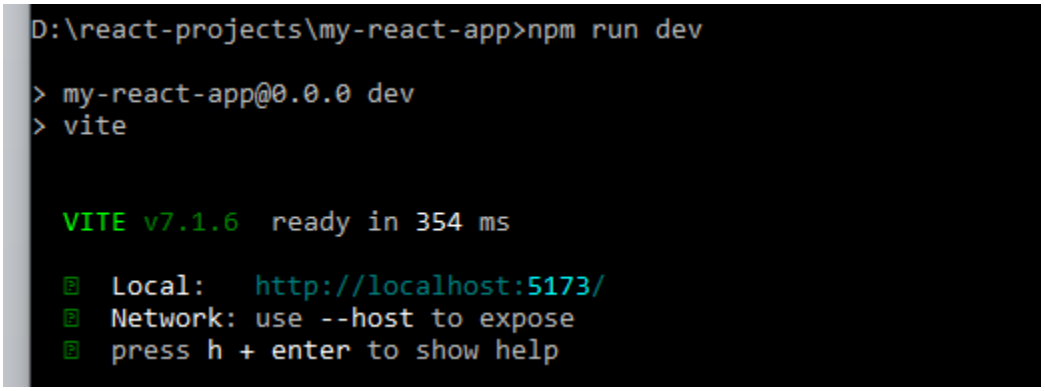
```
export default App;
```

`main.jsx` Entry Point

Make sure the app is rendering `<App />` in `main.jsx`:

```
// src/main.jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import './index.css'; // Optional

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Run it Command prompt:-



And now for

To Display **Multiple Pages** in your react app and navigate between them using **links**, you'll need to use **react router**, the most popular routing library for react.

□ Goal

You want to:

- Display multiple pages (e.g., Home and About)
- Navigate between them using links

□ Step-by-Step: Add Routing in React

□ Step 1: Install React Router

Run this in your project root (where `package.json` is):

```
npm install react-router-dom
```

□ Step 2: Create Pages

Create a folder `pages` inside `src`:

```
src/
├── App.jsx
├── main.jsx
├── components/
│   └── Greeting.jsx
└── pages/
    ├── Home.jsx
    └── About.jsx
```

`Home.jsx`

```
// src/pages/Home.jsx
import Greeting from '../components/Greeting';

function Home() {
  return (
    <div>
```

```
      <h2>Home Page</h2>
      <Greeting name="Alice" />
    </div>
  );
}

export default Home;
```

About.jsx

```
// src/pages/About.jsx
function About() {
  return (
    <div>
      <h2>About Page</h2>
      <p>This is the about page.</p>
    </div>
  );
}

export default About;
```

□ Step 3: Set Up Routing in `App.jsx`

```
// src/App.jsx
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-
dom';
import Home from './pages/Home';
import About from './pages/About';

function App() {
  return (
    <Router>
      <nav style={{ marginBottom: '1rem' }}>
        <Link to="/" style={{ marginRight: '1rem' }}>Home</Link>
        <Link to="/about">About</Link>
      </nav>

      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </Router>
  );
}

export default App;
```

□ Step 4: Ensure `main.jsx` is Correct

```
// src/main.jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

□ Result

Now you have:

- `/` → Home page (with `<Greeting />`)
- `/about` → About page
- Navigation links to switch between them

Let's add some **attractive CSS design** to your simple multi-page React app with routing.

We'll style:

- Navigation bar (header)
- Page layout (centered, padding, fonts)
- Buttons, links, and backgrounds

□ Design Style

We'll go with a **clean modern UI**, with:

- Centered content
- Light theme
- Rounded corners
- Hover effects

□ Folder Structure Overview

```
src/
├── App.jsx
├── main.jsx
├── index.css          ← global CSS
├── components/
│   └── Greeting.jsx
└── pages/
    ├── Home.jsx
    └── About.jsx
```

□ Step 1: Global Styles (`index.css`)

Edit or create `src/index.css`:

```
/* src/index.css */
body {
  margin: 0;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #f6f8fa;
  color: #333;
}

a {
  text-decoration: none;
  color: #555;
}

a:hover {
  color: #007acc;
}

nav {
  background-color: #ffffff;
```

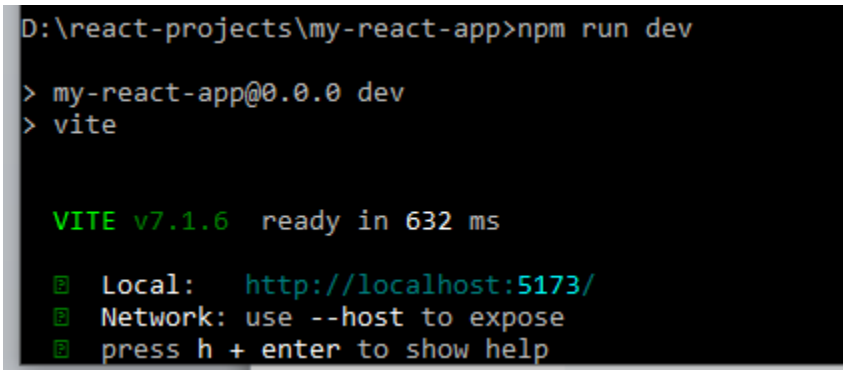
```
  padding: 1rem 2rem;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

nav a {
  margin-right: 1.5rem;
  font-weight: bold;
  font-size: 1.1rem;
}

.container {
  max-width: 800px;
  margin: 2rem auto;
  padding: 2rem;
  background-color: #ffffff;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.05);
}

h1, h2 {
  color: #007acc;
}

button {
  background-color: #007acc;
  color: white;
  border: none;
  padding: 10px 16px;
  border-radius: 5px;
  cursor: pointer;
  margin-top: 1rem;
  font-size: 1rem;
}

button:hover {
  background-color: #005f99;
}
```

□ Step 2: Update `App.jsx` with Class Names

```
// src/App.jsx
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-
dom';
import Home from './pages/Home';
import About from './pages/About';
import './index.css'; // Import global CSS

function App() {
  return (
    <Router>
      <nav>
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>
      </nav>
```

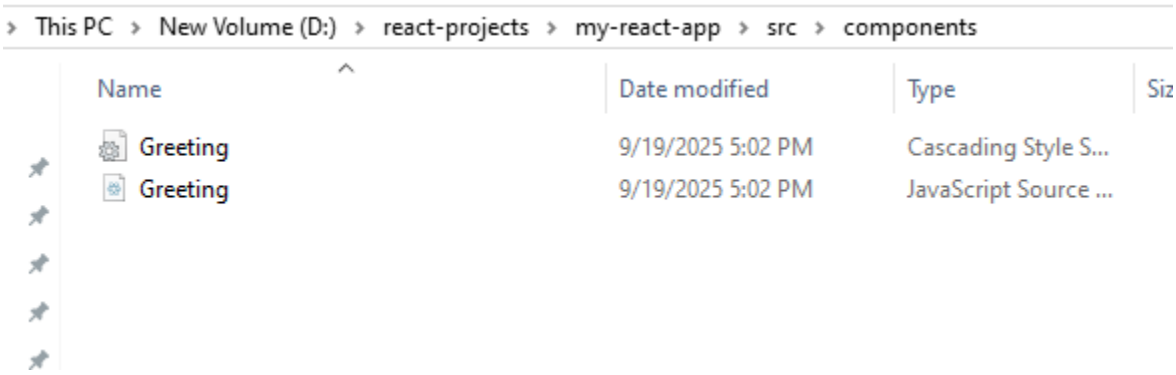
```
      <div className="container">
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

and now run and check :-



and Design your Greeting.jsx file :-

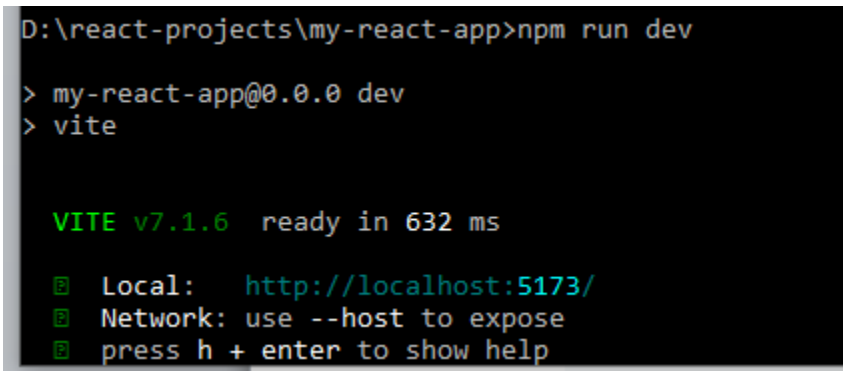
Create `Greeting.css inside src/components`

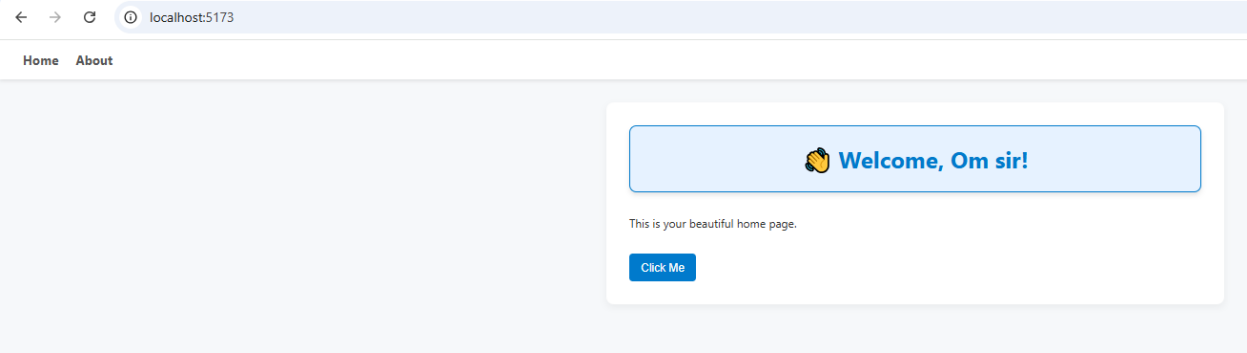


```css
/* src/components/Greeting.css */

.greeting-box {
  padding: 1.5rem 2rem;
  background-color: #e6f2ff;
  border: 2px solid #007acc;
  border-radius: 10px;
  text-align: center;
  margin-bottom: 2rem;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.greeting-text {
  font-size: 2rem;
  font-weight: bold;
  color: #007acc;
  margin: 0;
}
```

□ Update `Greeting.jsx`

```jsx
// src/components/Greeting.jsx
import './Greeting.css';

function Greeting({ name }) {
  return (
    <div className="greeting-box">
      <h1 className="greeting-text">□ Welcome, {name}!</h1>
    </div>
  );
}

export default Greeting;
```





Display Local Images (in `src/assets` folder)

✅ Step 1: Put the image in the project

Put your image (e.g., `avatar.png`) inside the `src/assets/` folder:

```
src/
├── assets/
│   └── avatar.png
├── components/
│   └── Greeting.jsx
```

✅ Step 2: Import and Use it

```
// src/components/Greeting.jsx
import './Greeting.css';
import avatar from '../assets/avatar.png';

function Greeting({ name }) {
  return (
    <div className="greeting-box">
      <img src={avatar} alt="User avatar" style={{ width: '100px',
borderRadius: '50%' }} />
      <h1 className="greeting-text">👋 Welcome, {name}!</h1>
    </div>
  );
}

export default Greeting;
```

✅ Result: You'll see the avatar image and the greeting message.

Note:-

what is advantage of it

npm create vite@latest my-react-app -- --template react

Great question! Let's break it down.

The command:

```
npm create vite@latest my-react-app -- --template react
```

is a fast and modern way to **create a new React project** using **Vite** instead of the older **Create React App (CRA)**.

□ What This Command Does

Part	Meaning
`npm create vite@latest`	Uses the latest version of Vite to scaffold a new project
`my-react-app`	The name of your new folder/project
`-- --template react`	Tells Vite to use the React template (JSX, `react` and `react-dom` preconfigured)

□ Advantages of Using `vite` Over Create React App (CRA)

Feature	Vite	CRA
□ **Speed**	Extremely fast dev server (cold start in ms)	Slower (uses Webpack)
□ **Modern Tooling**	Uses ESBuild & Rollup (faster, modern bundler)	Uses Webpack (heavier)
□ **Hot Reload**	Instant updates (HMR)	Slower refreshes
□ **Simpler Config**	Minimal config out of the box	Heavier config by default
□ **Lightweight**	Less boilerplate	More files and dependencies
□ **Package Optimization**	Auto-optimizes dependencies	No built-in optimization
□ **Plugin Ecosystem**	Rich plugin system (like Rollup)	Webpack-specific
□ **Native ESM Support**	Yes (browsers & dev tools)	No

□ After Running the Command

Once you run:

```
npm create vite@latest my-react-app -- --template react
```

You will:

1. Get a **clean React project** using Vite
2. Be prompted to install dependencies:

```
cd my-react-app
npm install
npm run dev
```

Then visit: `http://localhost:5173` (Vite default port)

□ Example Folder Structure

```
my-react-app/
├── index.html
├── package.json
├── vite.config.js
└── src/
    ├── main.jsx
    └── App.jsx
```

Very minimal and clean.

□ When Should You Use It?

Use Vite when:

- You're starting a **new React project**
- You want **fast development experience**
- You want to avoid **heavy configs and slow builds**
- You plan to use **modern JavaScript and tooling**

□ When You Might Avoid It

You **might stick to CRA** if:

- You're maintaining an older CRA-based codebase
- Your team is heavily invested in Webpack-specific tools

But for new projects, **Vite is the better choice** in 2025.

□ TL;DR

```
npm create vite@latest my-react-app -- --template react
```

□ Creates a blazing fast, modern React project with Vite
□ Faster than Create React App
□ Ideal for modern web development