Create React app

Creating a vite project	`npx create-react-app dojo-blog`
	`cd to new pakige`
	`npm start`
Install all depandesy (node-moduls folder – need if pushed from GitHub)	`npm install`
Instull pacige to VS code	`Simple React Snippets`
Create stateless functional component	`sfc + tab`
Quick nav className="navbar">	`nav.navbar + enter`
duplicate a row in VS Code, you can use a simple keyboard shortcut:	`Shift + Alt + Down Arrow`

Chrome instanton for React https://chromewebstore.google.com/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi	`react developer tools`
To get text in ant tag	`Lorem8`

used to set up a local ***REST API server*** using a JSON file (data-db.json) as the ***data source***. Here's what each part of the command means:	`npx json-server --watch data/db.json --port 8000`
	~~`http://localhost:8000/products`~~
	`http://localhost:8000/blogs`

`json-server` essentially creates a **fake (mock) database** using a JSON file. It mimics a real database by providing a fully functional REST API that you can use to perform operations like `GET, POST, PUT,` and `DELETE` requests. The data in your JSON file (`data-db.json`) acts as the source of truth, allowing you to quickly simulate how your application would behave when interacting with a back-end server.

The React Router - is primarily a library that implements a routing pattern for React applications. Right now, our React app only has one page, and we don't have a way to navigate to different pages like most websites. But in most real-world websites, there are usually multiple pages (like "Home," "About," "Contact"), and we need to be able to move between them. In React, we can do this using something called **React Router**.

How Websites Normally Work (Without React)

In a regular website (without React), when you type a website address in your browser (like `example.com`), your browser sends a request to a server, and the server sends back an **HTML page**. If you click on a link to go to another page (like `example.com/contact`), the browser sends another request, and the server sends back a **new HTML page** for the Contact page. Each time you click on a link, a request is made, and the server responds with a new page.

How React and React Router Work

In a **React** app, things work a little differently. When you first visit the website, the server sends back a ***blank or nearly blank HTML page***, along with some **React JavaScript files**. These files control everything that happens on the page.

After this initial load, React takes over. So when you click on a link to a different page (like "Contact"), instead of going back to the server to get a new page, **React Router** steps in and says, "No need to ask the server for a new page. I'll just change the content on the page using JavaScript." This makes everything **faster** because you're not constantly loading full pages from the server. React just updates part of the page, like the main content area, with the new information.

How We Add React Router to Our App

1. **Install React Router:** To use this feature, we first need to install React Router by running the following command:

```
npm install react-router-dom@5
```

(We're using version 5 because it's stable. There's a newer version, but it's still in development.)

2. **Set Up the Router in Your Code:** Once installed, we need to set up React Router in our code. First, we import it like this:

```
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
```

3. **Wrap the App with Router:** We then wrap the whole app with `<Router>` so that React Router knows it should handle all the routing (page changes):

```
<Router>
  {/* Your entire app goes here */}
</Router>
```

4. **Setting Up Routes:** Now, we define routes for each page we want. For example, for the homepage, we would create a route like this:

```
<Route path="/" component={Home} />
```

This tells React Router: "When the user visits the **/ (homepage),** show the Home component."

5. **Using Switch:** We put all the routes inside a `<Switch>` component. This ensures that only one route is shown at a time:

```
<Switch>
  <Route path="/" component={Home} />
  {/* More routes can go here */}
</Switch>
```

What Happens Behind the Scenes

- When you click a link (like "Contact"), instead of asking the server for a new HTML page, React Router steps in and says, "I'll handle this," and then **injects** the correct content for the "Contact" page without reloading the whole page.

Why Is This Good?

1. **Faster**: Since we're not reloading the entire page, the app feels faster.
2. **Smoother**: Transitions between pages feel more seamless because everything is handled in the browser.

3. **Less Server Load**: The server doesn't need to send new HTML pages all the time

When you first load a React app, the server typically sends an initial HTML file along with a bundled JavaScript file (or multiple files if code splitting is used). This bundle ***includes all the JavaScript*** needed to run your React application, which also includes React Router if you're using it.

However, while the initial JavaScript bundle can be large, many modern React apps use techniques like **code splitting** to break the app into smaller chunks that are only loaded when needed, meaning not all of the JavaScript is sent at once on the first request. Only the necessary parts of the app are loaded, and additional pieces (such as code for other routes) are loaded on-demand when the user navigates to different sections of the app.

To install React Router in your React projec	`npm install react-router-dom@5` (5 – version)
When we unstull something it goise4 to	`node_modules` folder

What the Code Does

This code sets up a basic React app that lets users navigate between different parts of the app using links. It uses a tool called **React Router** for this navigation.

Key Parts Explained

1. **Imports:**
   - **Navbar** and **Home:** These are different parts of your app. The **Navbar** usually has links to other pages, and **Home** is the main page shown when you first visit the app.
   - **Router, Route, Routes:** These are special tools from React Router that help manage navigation in your app.
2. **Router:**
   - The `<Router>` wraps everything in your app. It keeps track of where you are in the app (like what page you're on). This is important for allowing you to switch between different pages.
3. **Navbar:**
   - The `<Navbar />` is displayed all the time. It has links (like Home, About, etc.) that users can click to go to different parts of the app.
4. **Routes:**
   - The `<Routes>` is where you define what happens when someone visits a specific URL in your app. It checks the URL and shows the right page.
5. **Route:**

- The `<Route>` tells the app which component (page) to show based on the URL. For example:
  `<Route path="/" element={<Home />} />`
  This means that when someone visits the main page (the URL /), the **Home** component will be displayed.

6. **Content Area:**
   - The `<div className="content">` holds the parts of the app that change when you navigate to different pages.

Adding More Pages

If you want to add another page, like an About page, you can do it like this:

1. Create a new **About** component.
2. Add a new route inside the `<Routes>` like this:
   `<Route path="/about" element={<About />} />`

This means that when someone goes to `/about,` the **About** page will show up.

Summary

In simple terms, this code helps you create an app where users can click links to move between different pages. The **Router** keeps everything organized, the **Navbar** shows the links, and **Routes** and **Route** manage which page to show based on the URL.