**React**

HOWEVER, if you really want to develop locally using **VSCode** then you can follow the steps below to work on the module locally. You will need to adapt to some differences between your code and what you see in the videos. But each lesson has downloadable .zip files for you to follow along.

**Step 1**. Make sure you have the latest version of Node installed. If not head over to <https://nodejs.org/en/download> to download the LTS (Long Term Support) version of Node.

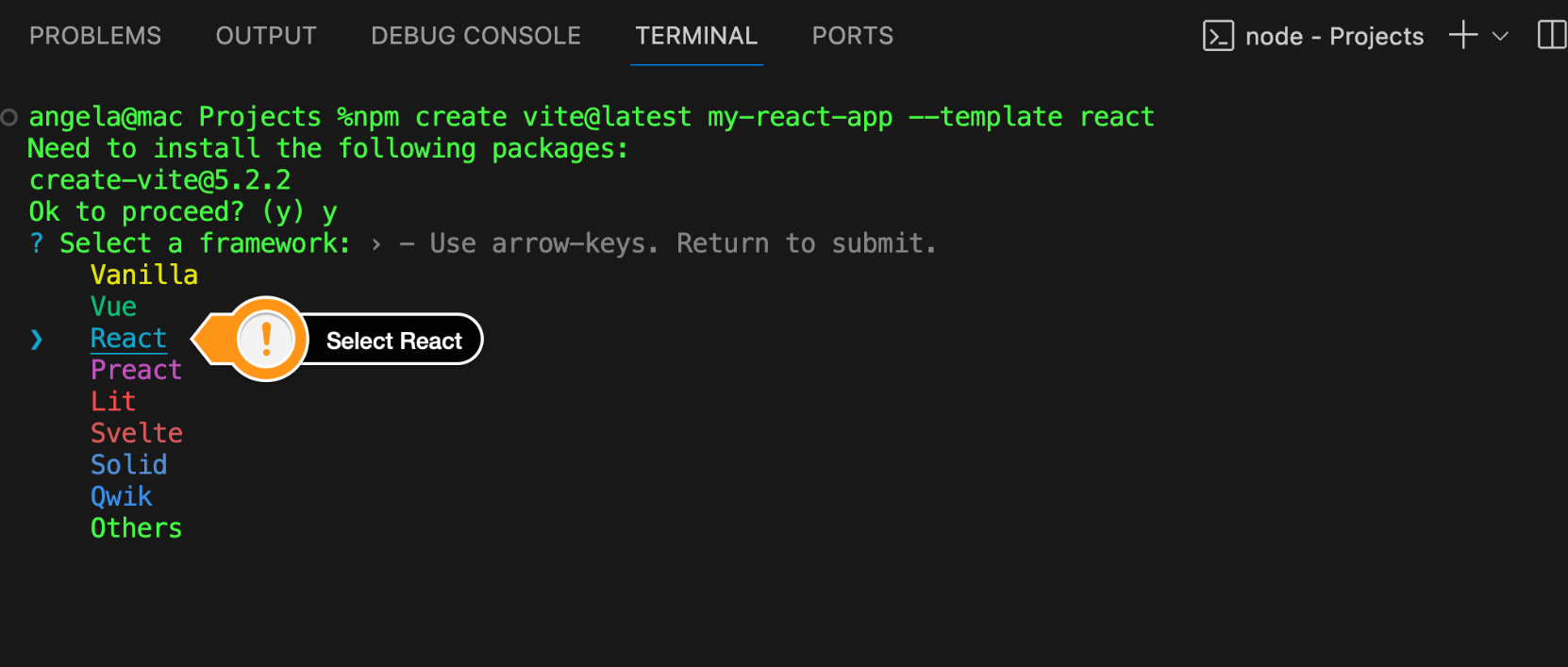
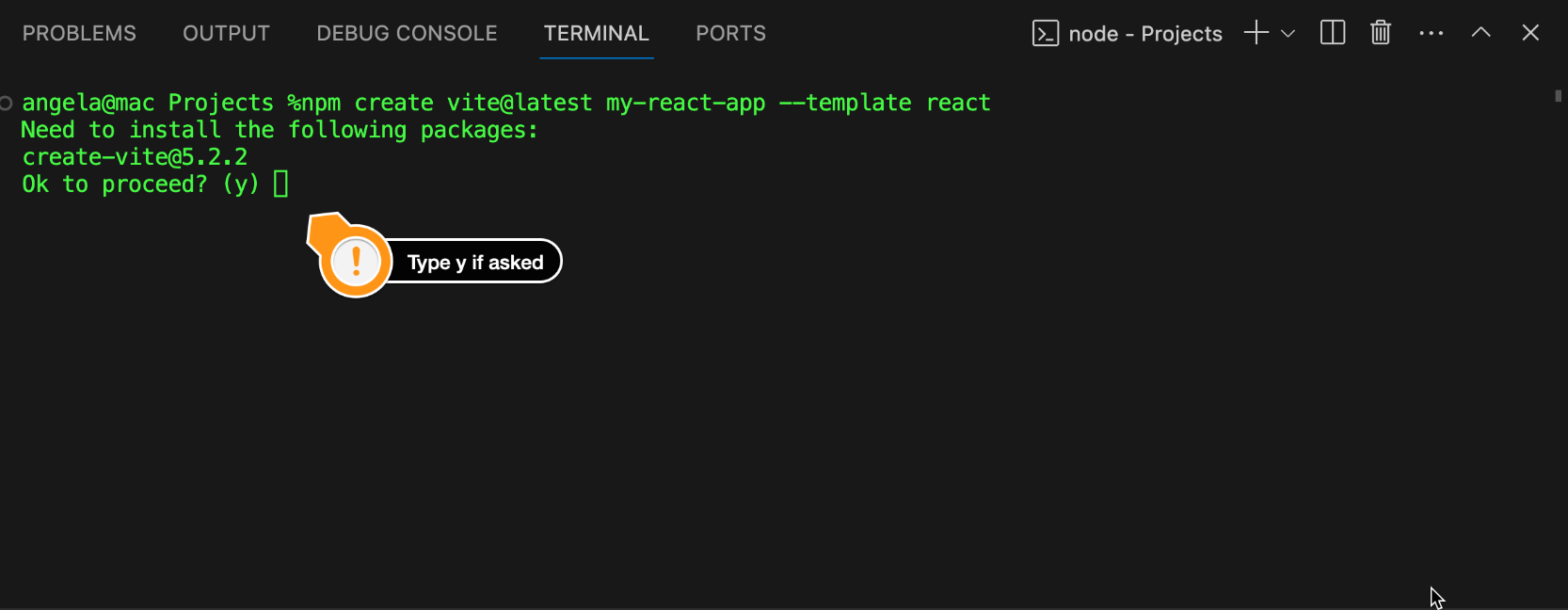
**Step 2**. Make sure you have the latest version of VSCode installed. If not, head over to <https://code.visualstudio.com/download> to download the version for your platform.

**Step 3**. Open a Terminal or command prompt and navigate to the directory where you want to create your React project.

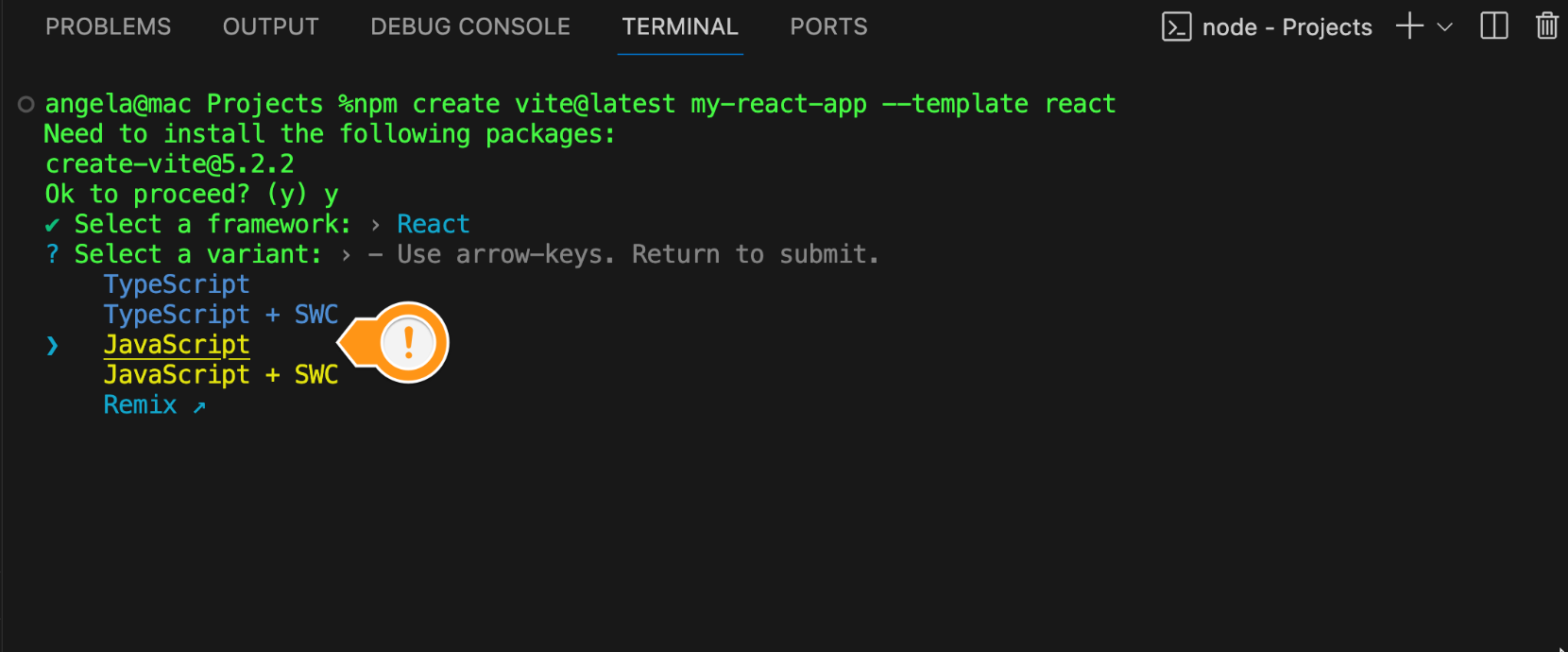
**Step 4**. Create a Vite app by running the following command in your Terminal or Command Prompt:

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

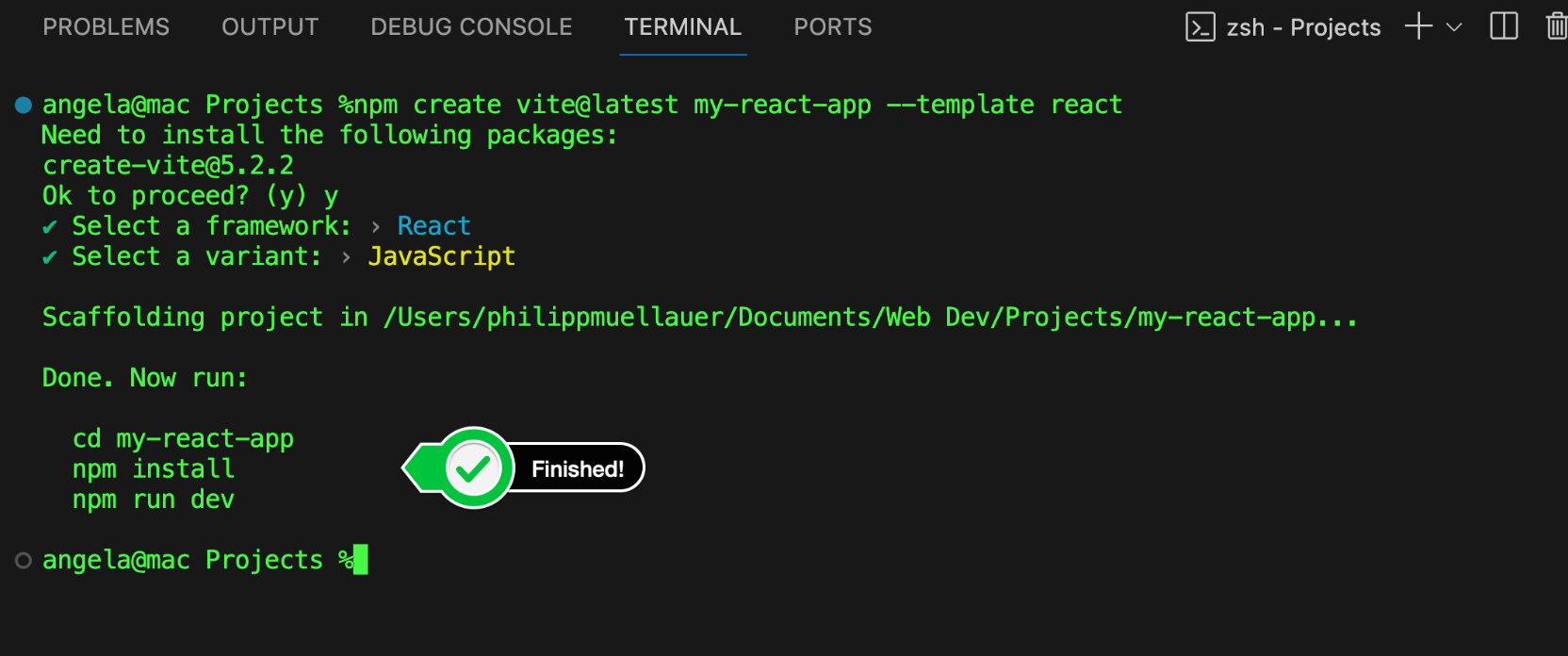
**Step 5**. The first time, you won't have Vite installed. Type y to proceed. Then you'll be asked to select a framework. Use your down arrow to select React.



**Step 6**. You'll be asked to select a variant, select Javascript.



Then wait for the installation to finish, this will take a few minutes.

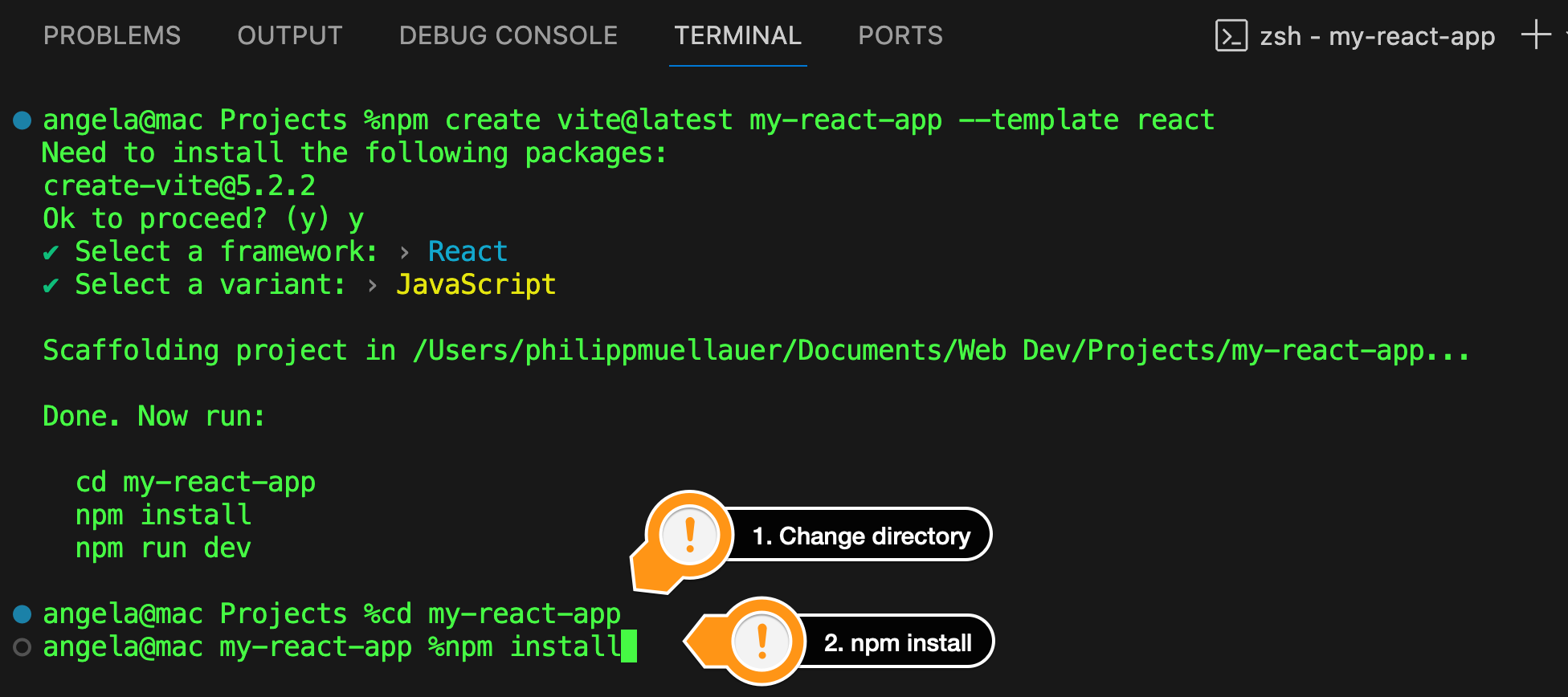


**Step 7**. Change directory to the new app that you built.

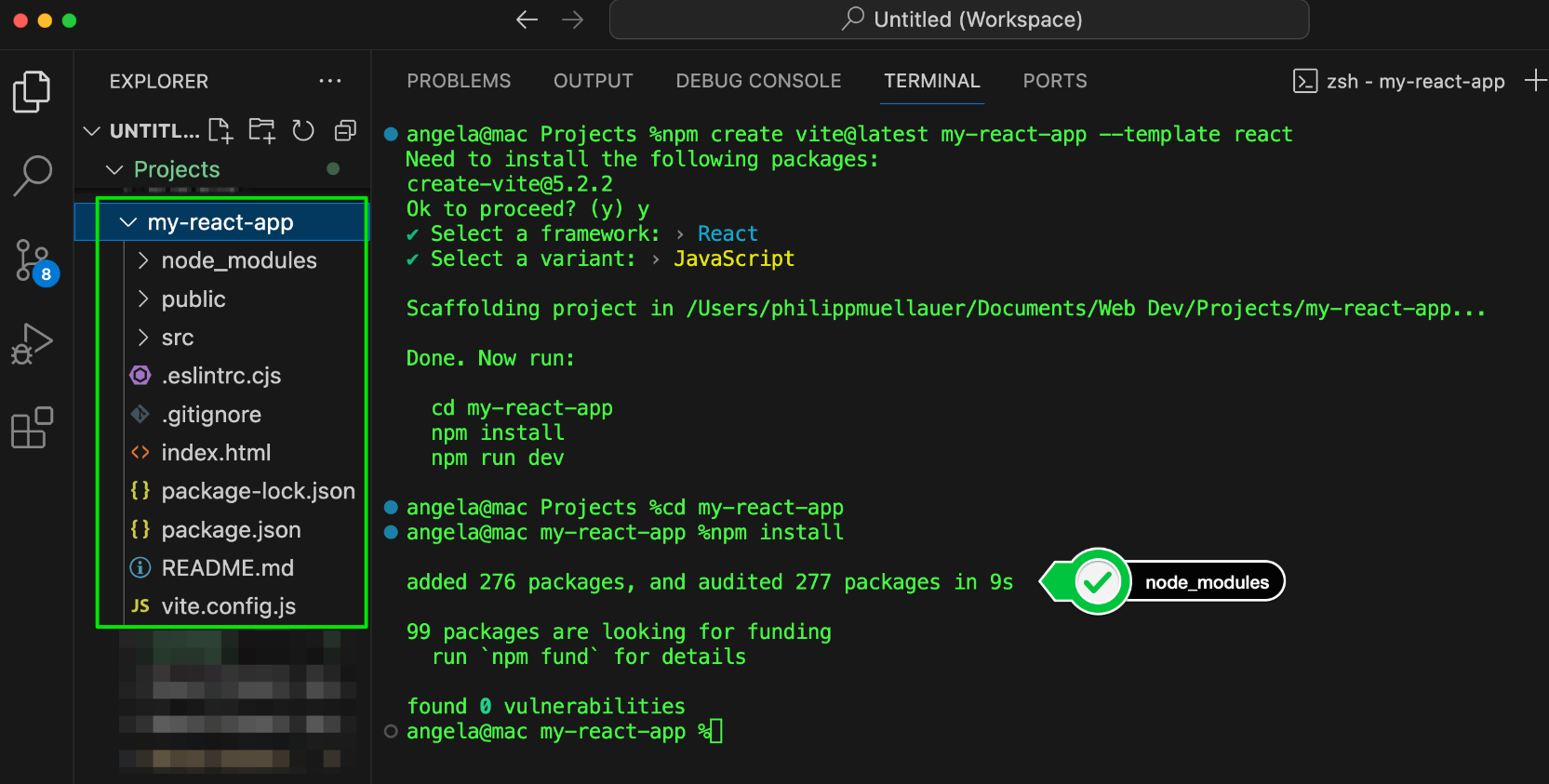
cd my-react-app

**Step 8**. Install dependencies:

npm install



When npm has installed all the necessary packages, open your project folder in VS Code. You should see a node modules folder:



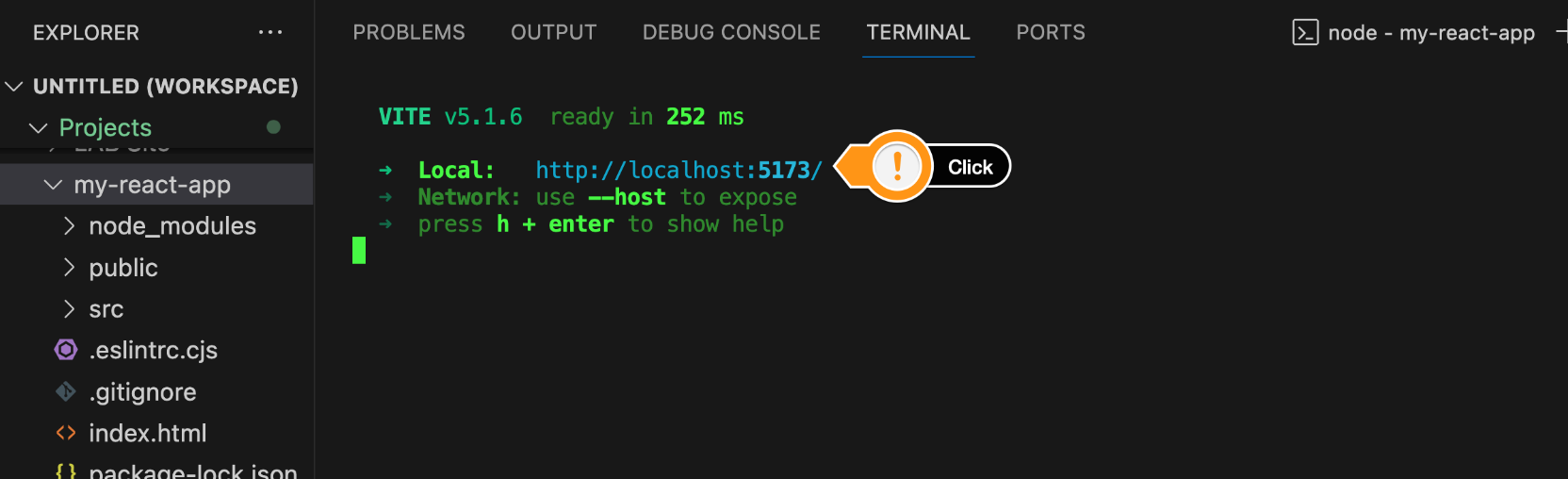
**Step 9**. Start the development server:

npm run dev

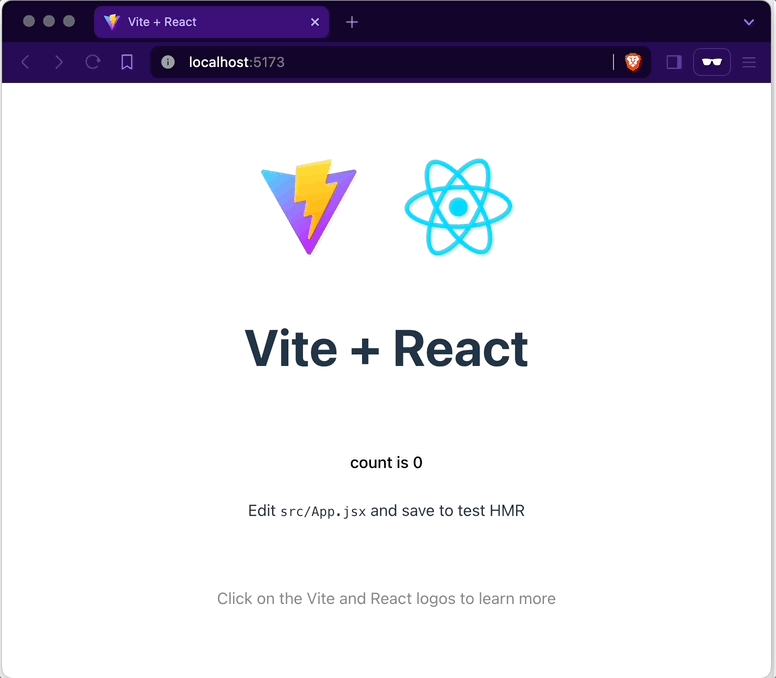
npm run dev

Vite will compile your code every time your change anything and you can see the location of your development server in the output.

Step 10. Open the app in your browser by heading over to the local address shown. It's usually at http://localhost:5173/



You can now work on this local version and Vite will automatically recompile your code as you make changes.



We can also use g<https://codesandbox.io/> to build React App

**JSX :**

React Components Overview

In React, components are reusable, independent building blocks that define the UI of an application.

Components can be functional (stateless) or class-based (stateful), and they help in maintaining a modular and scalable code structure by managing UI logic efficiently.

Why Use Components?

✅ Reusability – Write once, use multiple times.

✅ Maintainability – Easier to manage and update.

✅ Code Organization – Breaks UI into smaller, manageable parts.

**Types of Components in React**

* **Functional Components (Recommended)**
* **Class Components (Older Approach)**

Functional Components

* Written as JavaScript functions.
* Uses React Hooks for state and lifecycle management.
* Simple and easy to read.

Example of Functional Component:

function Greeting() {

return <h1>Hello, Welcome to React!</h1>;

}

export default Greeting;

Class Components

* Uses ES6 class syntax.
* Manages state using this.state.
* Requires render() method.

Example of Class Component:

import React, { Component } from "react";

class Greeting extends Component {

render() {

return <h1>Hello, Welcome to React!</h1>;

}

}

export default Greeting;

A **class component** in React is a JavaScript class that extends React.Component and must have a render() method to return JSX. It supports **state** and **lifecycle methods**, making it useful for managing complex logic.

**Example:**

javascript

CopyEdit

import React, { Component } from 'react';

class MyComponent extends Component {

state = { message: "Hello, World!" };

render() {

return <h1>{this.state.message}</h1>;

}

}

export default MyComponent;

**Key Points:**

* Supports **state** and **lifecycle methods**.
* Requires a render() method to return UI.
* Less common now, replaced by **function components with hooks**.

This knowledge is essential for working with older React codebases.