1. React JS Introduction

React is an open source, JavaScript library for developing user interface (UI) in web application. React is developed and released by Facebook. Facebook is continuously working on the React library and enhancing it by fixing bugs and introducing new features.

How does React Work?

React creates a VIRTUAL DOM in memory.

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

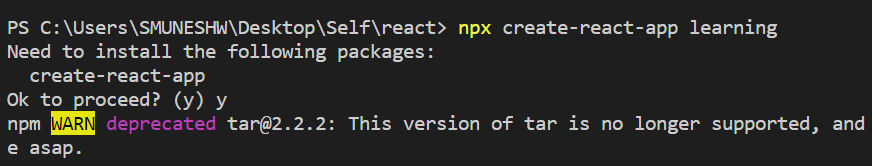
React only changes what needs to be changed!

To generate react project

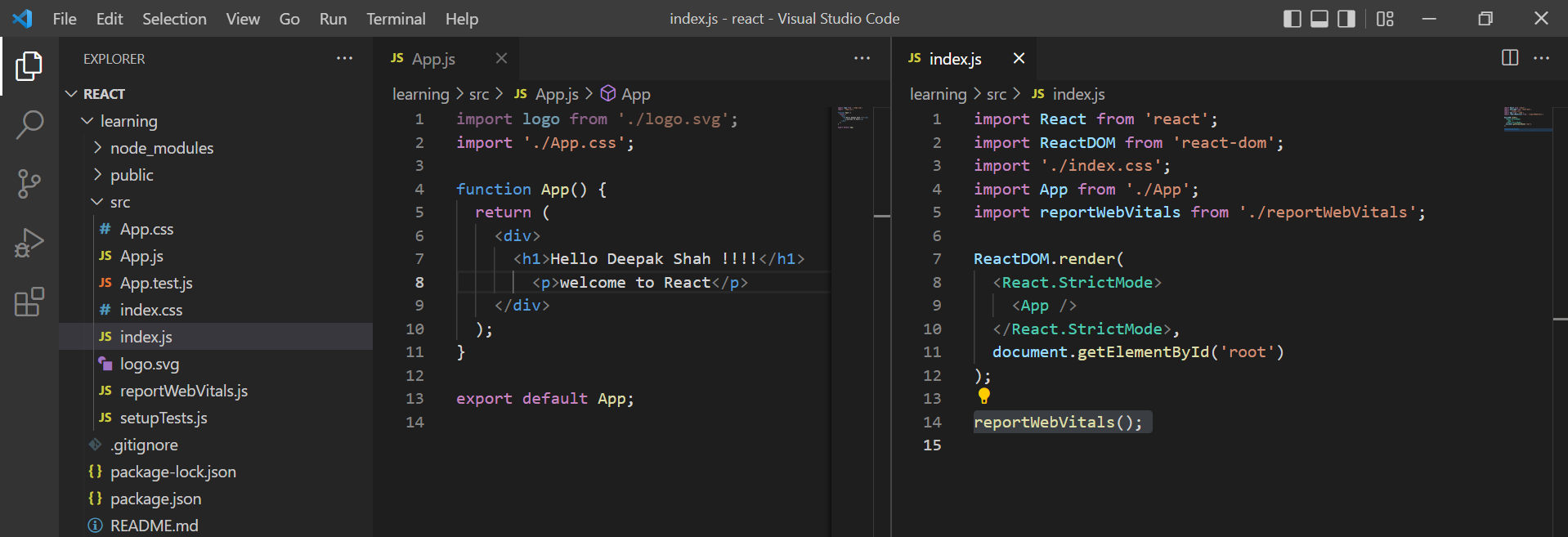
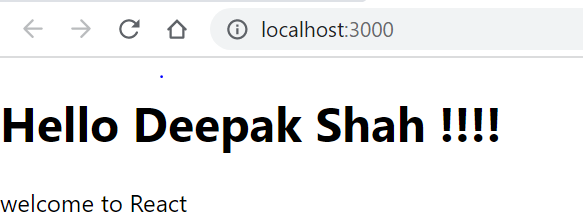
npx create-react-app my-app

To run the react project

npm start



Hello program

Whenever program runs the index.js page first get load and then later app.js

Package.json file contains the version of react and package-lock.json contains the version history of files which are added.

**Interview:** Alternative of npm is Yarn package manager (YPM) developed by fb.

What is JSX?

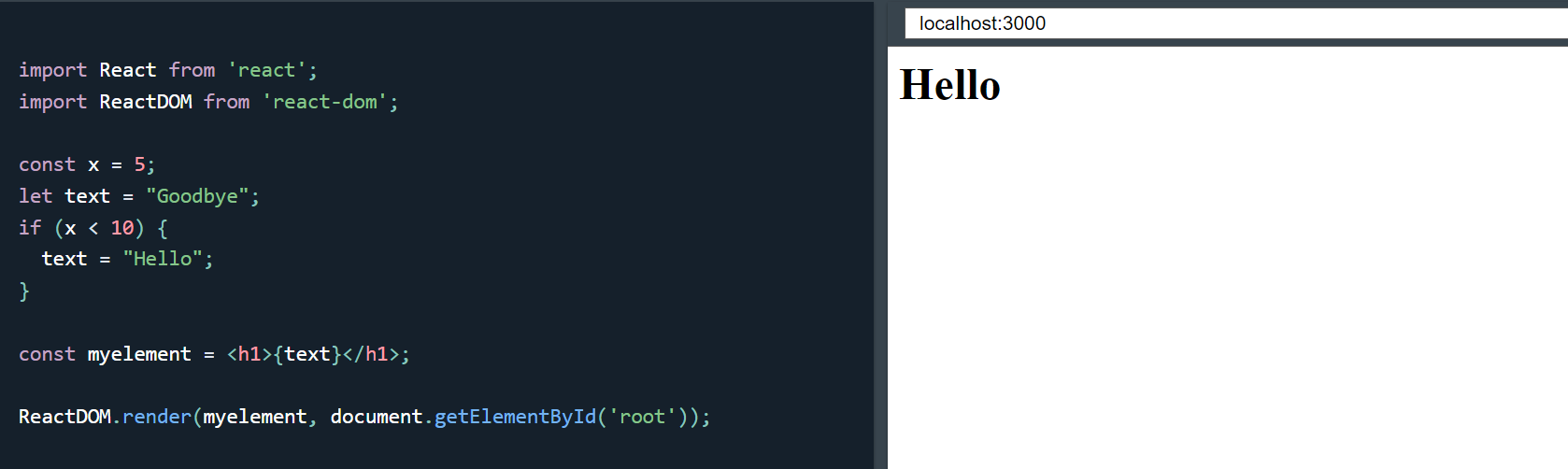
JSX stands for JavaScript XML.

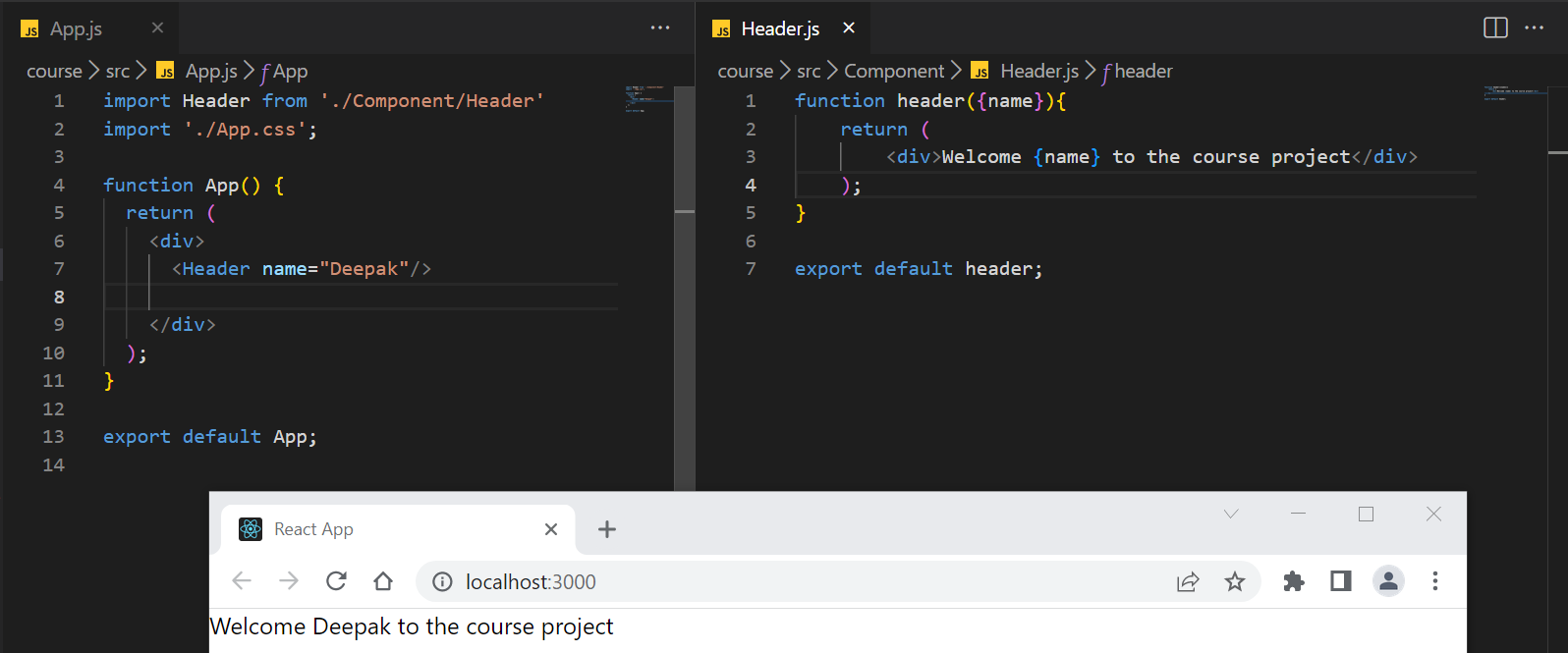
JSX allows us to write HTML in React.

JSX makes it easier to write and add HTML in React.

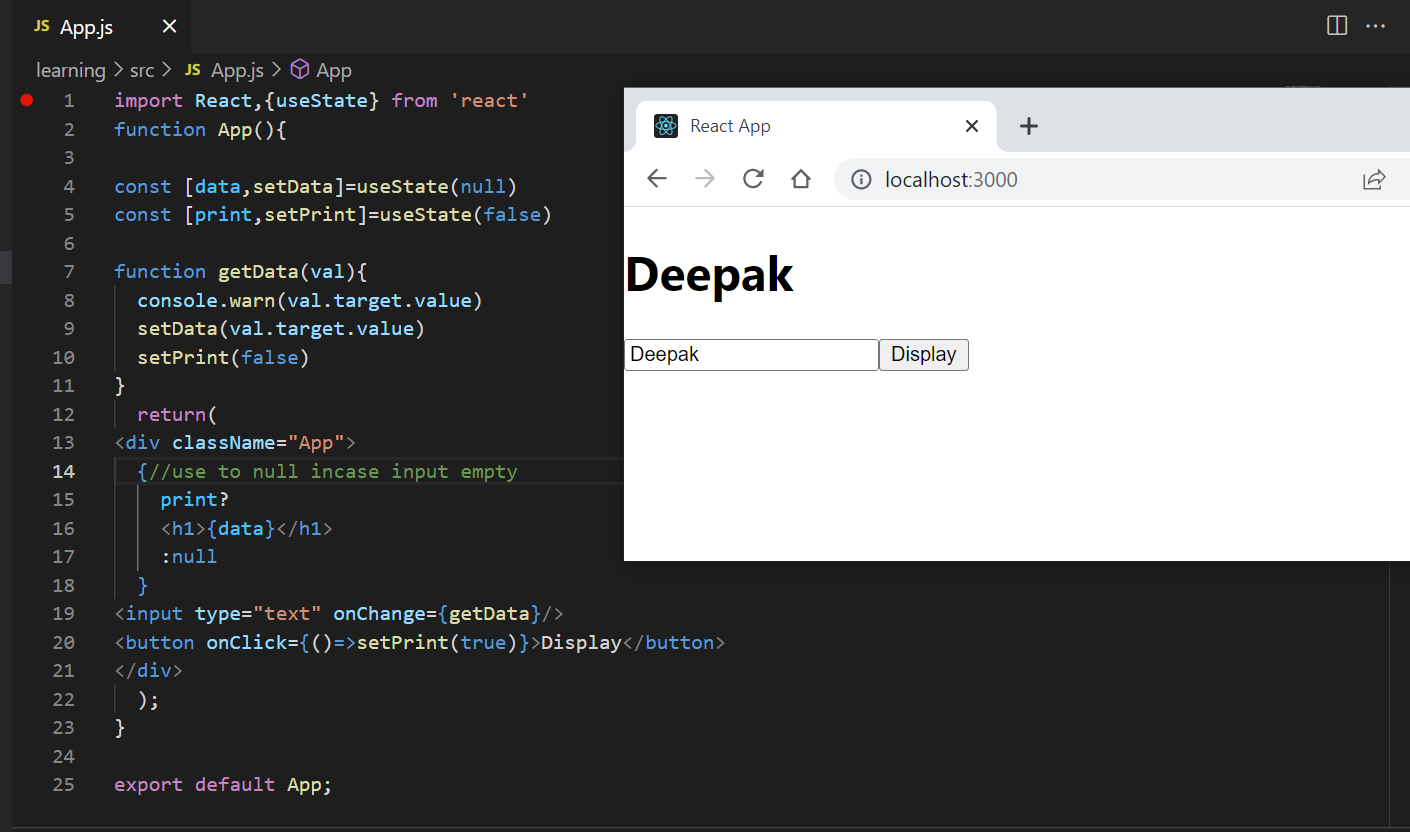
These brackets { } are know as jsx brackets

Interpolation:





Get Input type text value:



1. Components

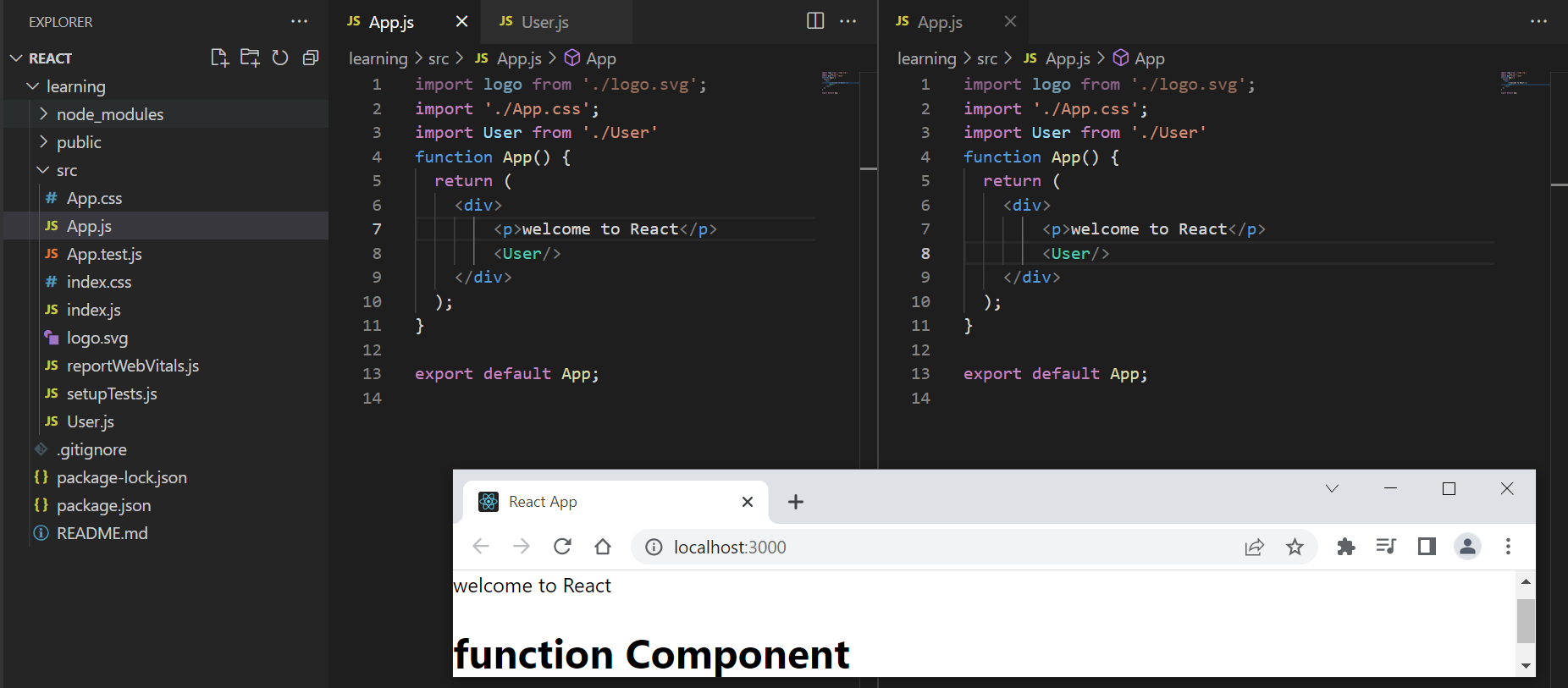
Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.

Header and footer are example of component

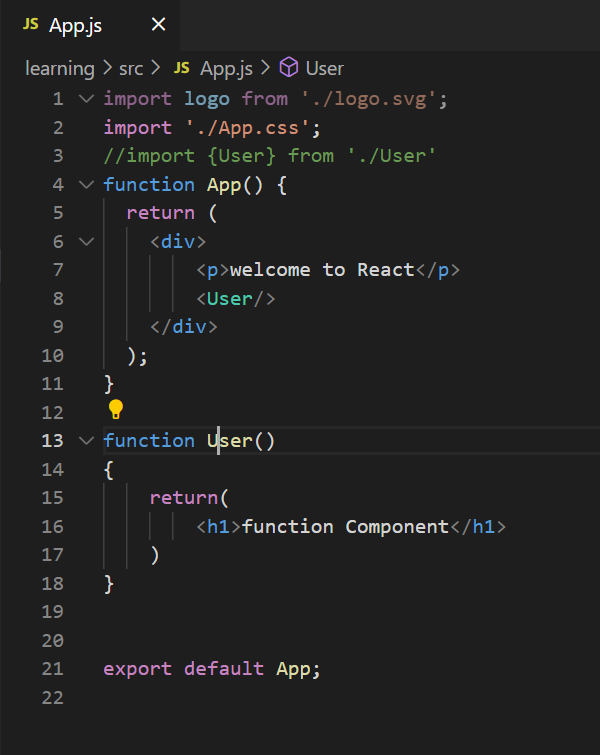
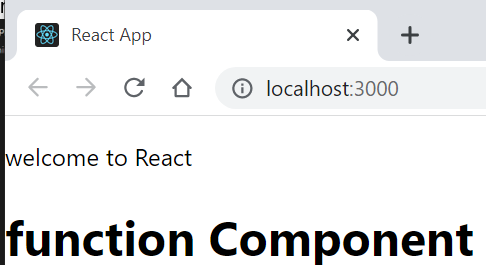
Components come in two types, Class components and Functional components,

1. **Functional Component**

A Functional component also returns HTML and behaves much the same way as a Class component, but Functional components can be written using much less code, are easier to understand.



Alternate approach

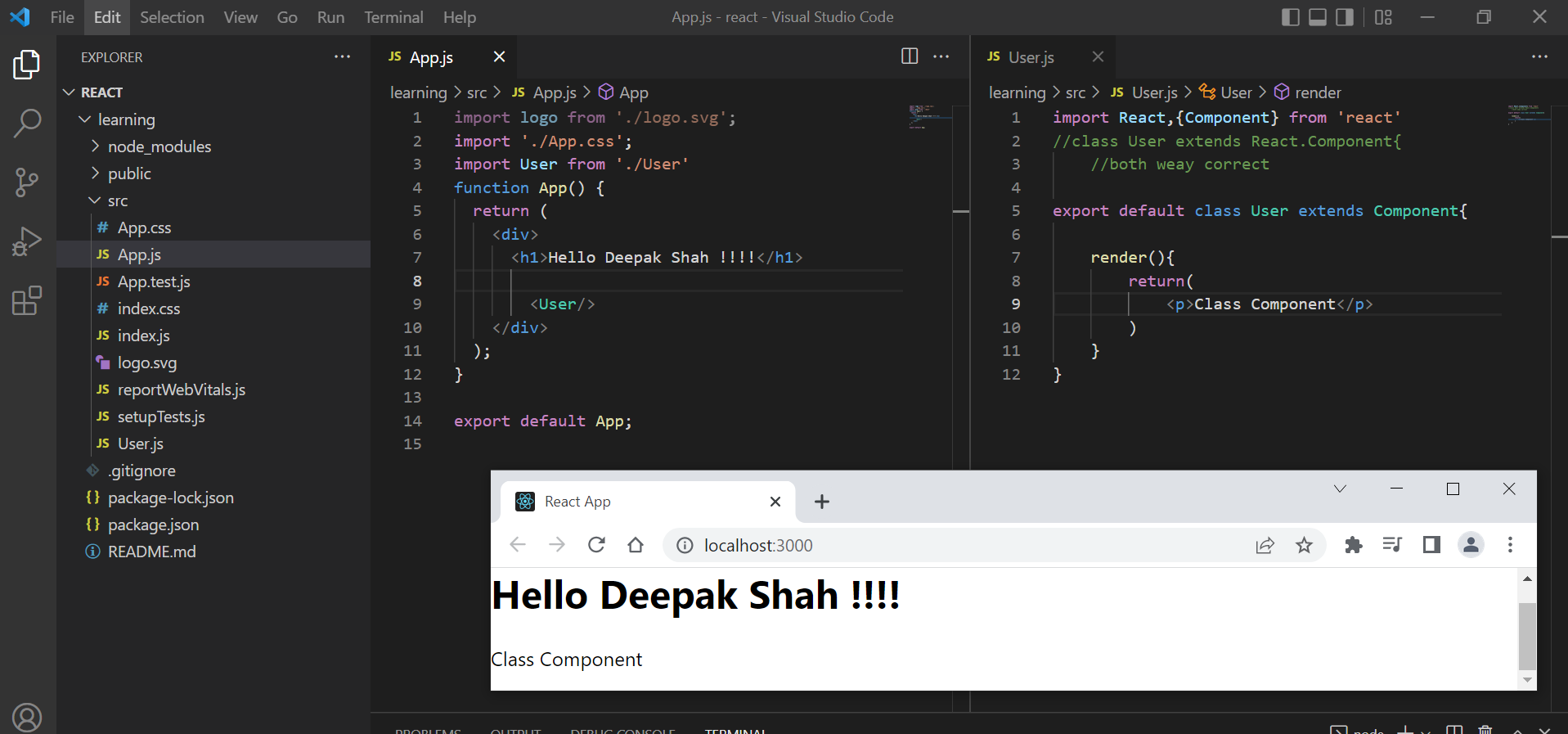
 

***Use function component only in your code as class component may deprecate***

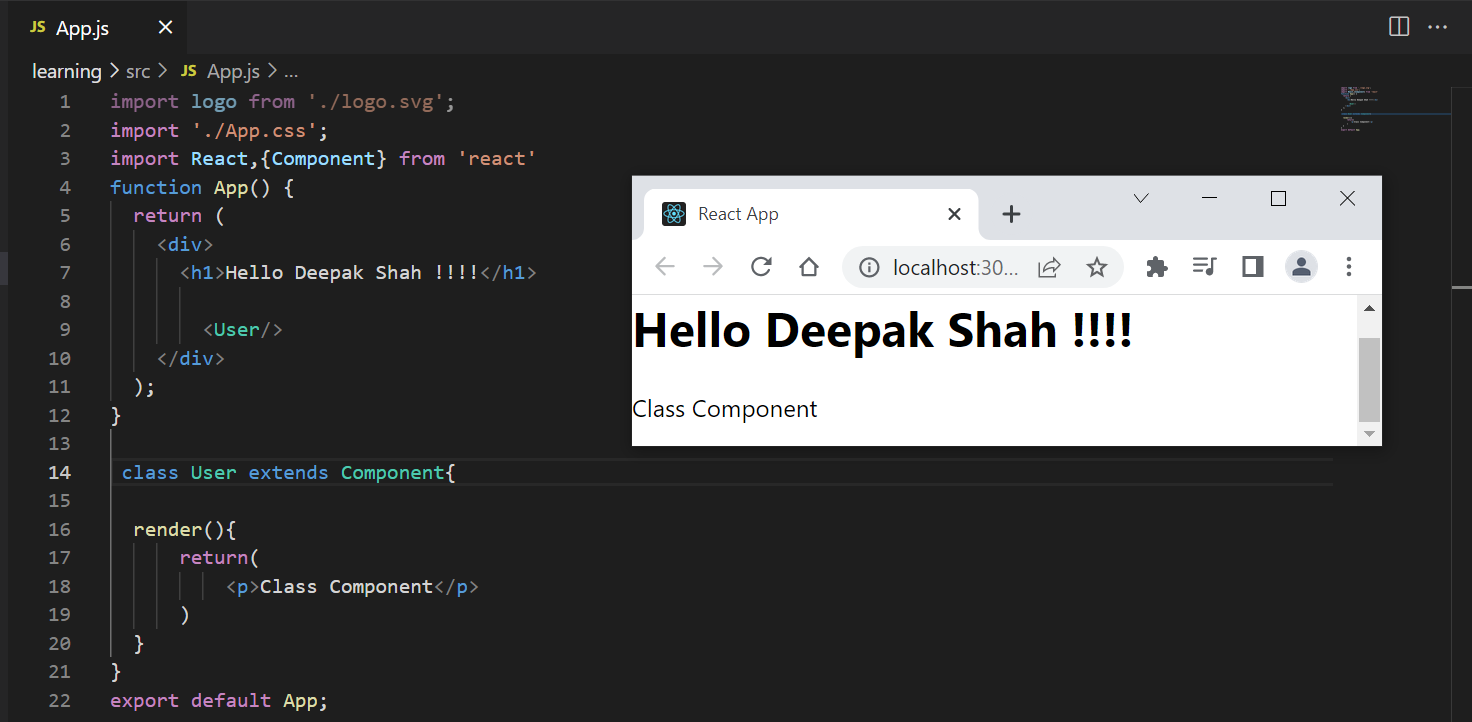
1. **Class Component**

A class component must include the extends React.Component statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a render() method, this method returns HTML.



Alternate approach (Not prefer)



**Interview:** Can we create a function inside another function or in app function.  
**Answer :** Yes we can example  
function app()

{

function apple(return <div>Deepak</div>){}

</apple>

}

**2.a React State (Functional and Class)**

React components has a built-in state object.

The state object is where you store property values that belongs to the component.

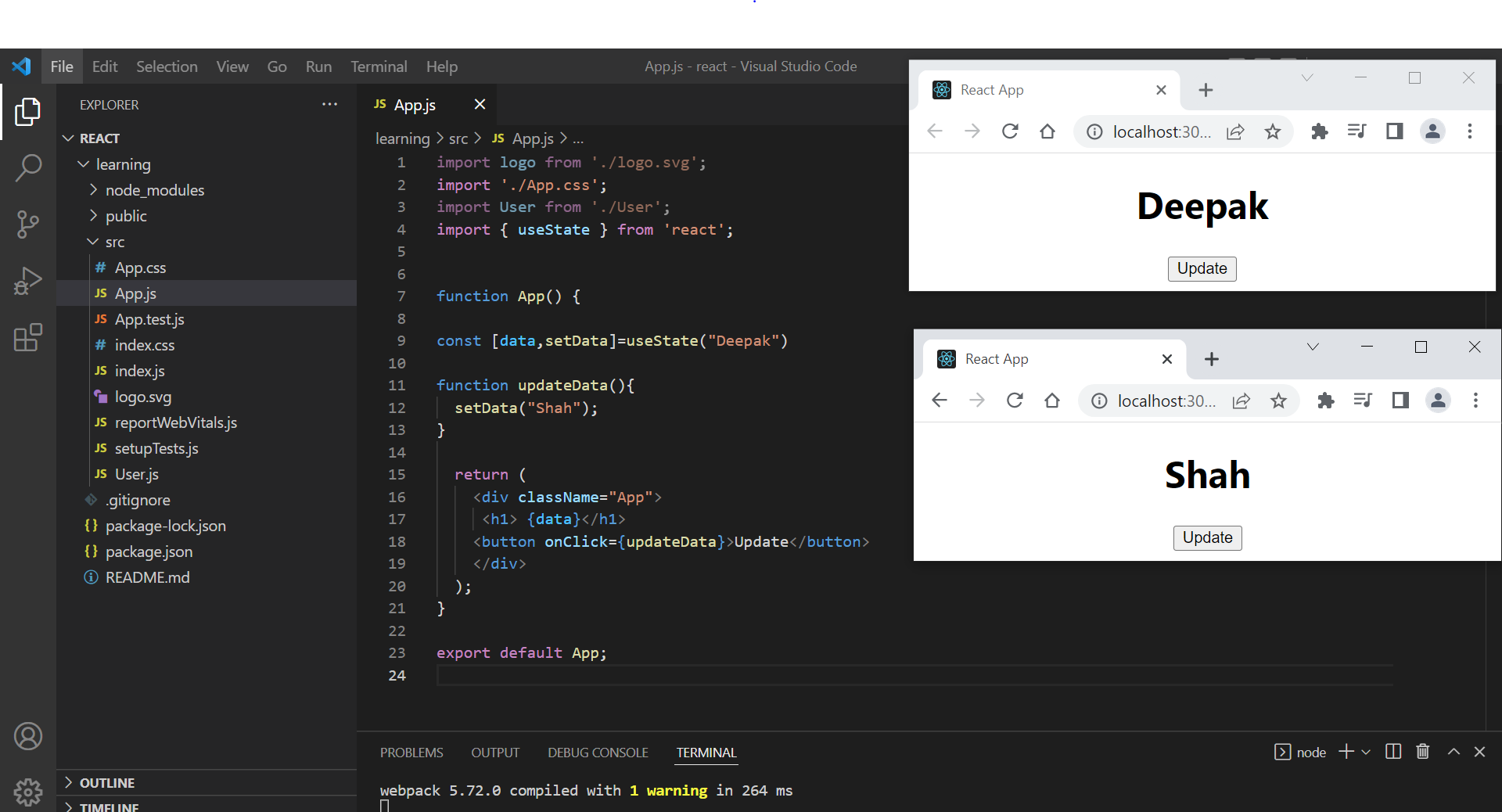
When the state object changes, the component re-renders.

The state object is initialized in the constructor:

Why we need state?

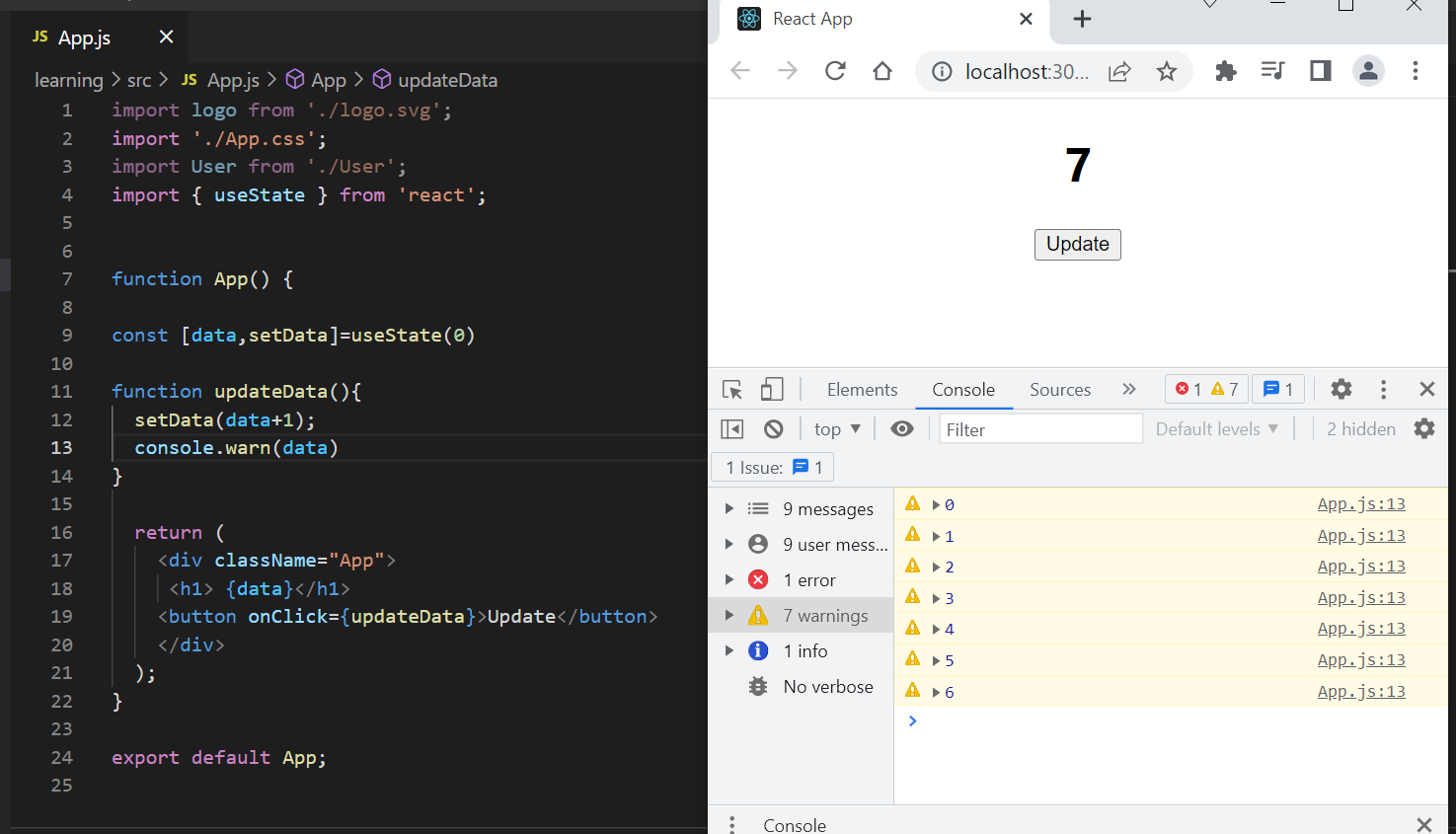
* State is an object are used as a variable to store properties while using variable or let components are not updating hence, we use state in react. **State is public**

1. State with functional component

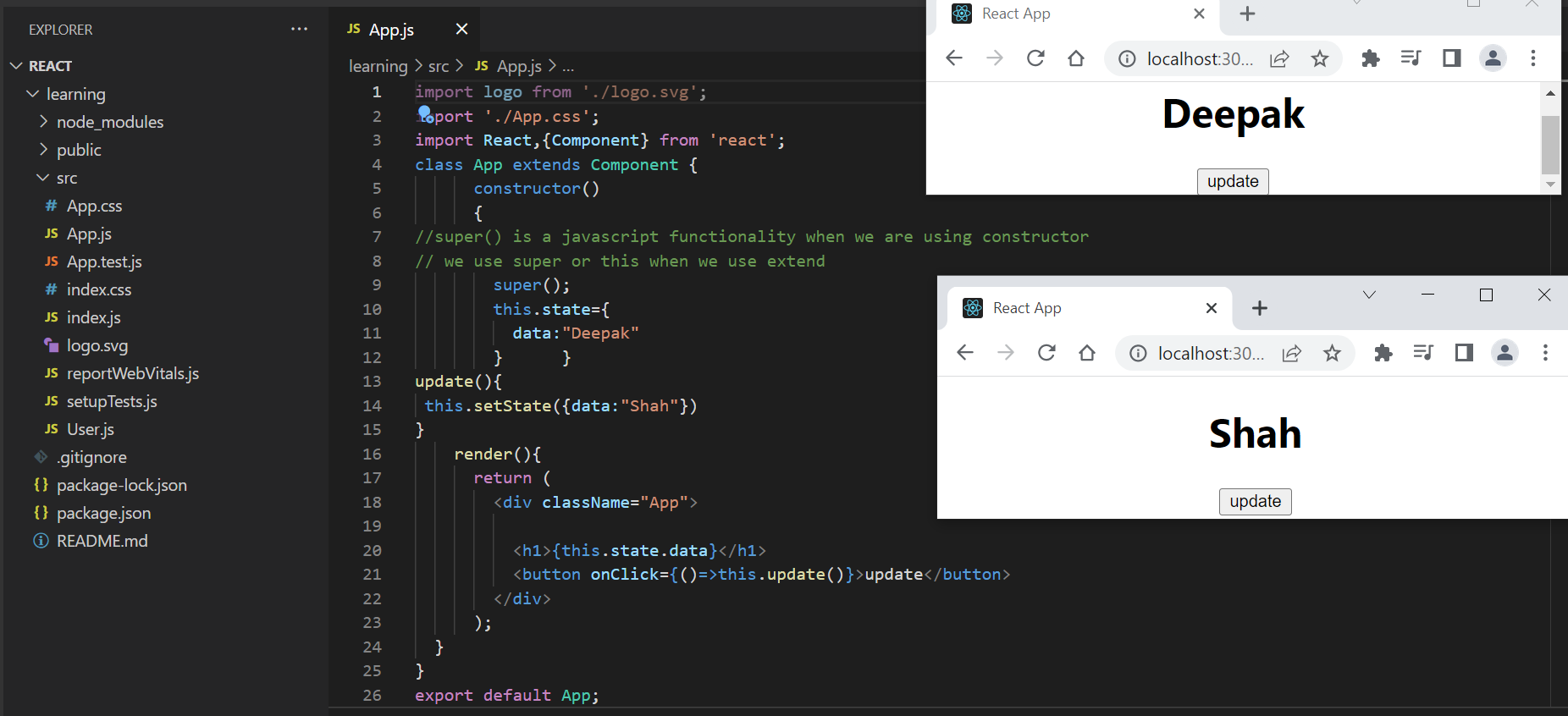


*Before and after click*

Example 2

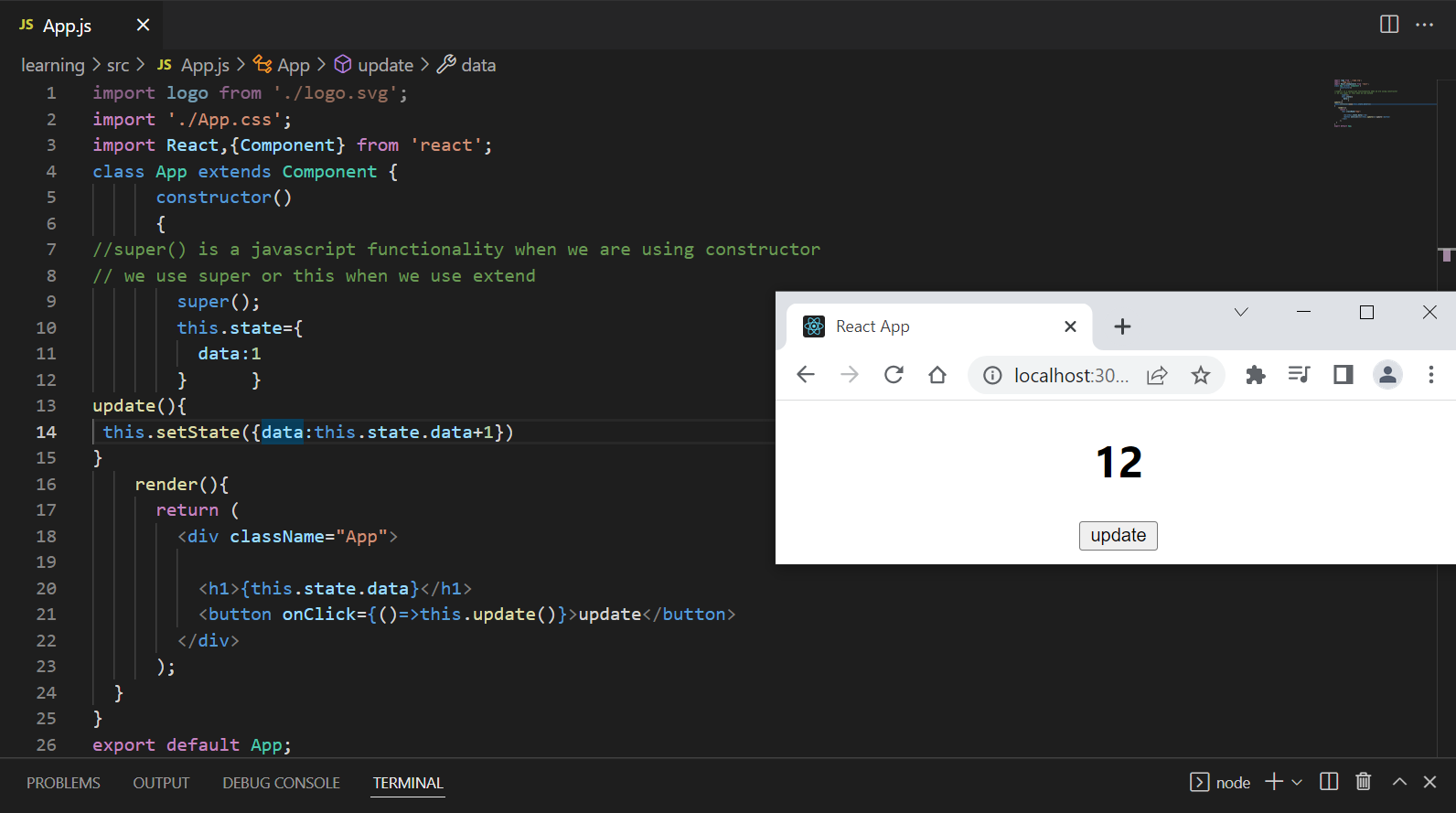


1. State with class component



*Before and after click*

Example 2

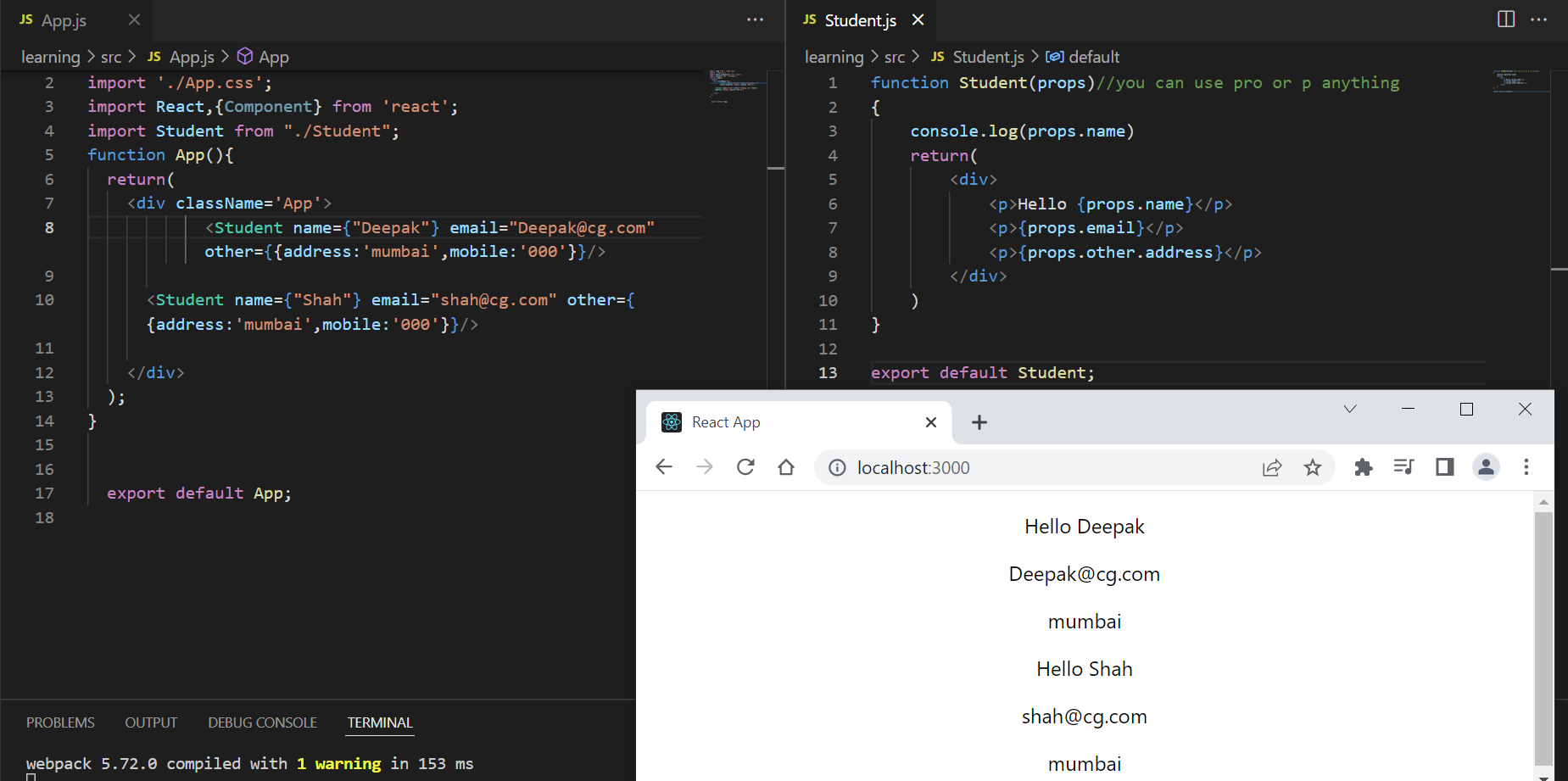


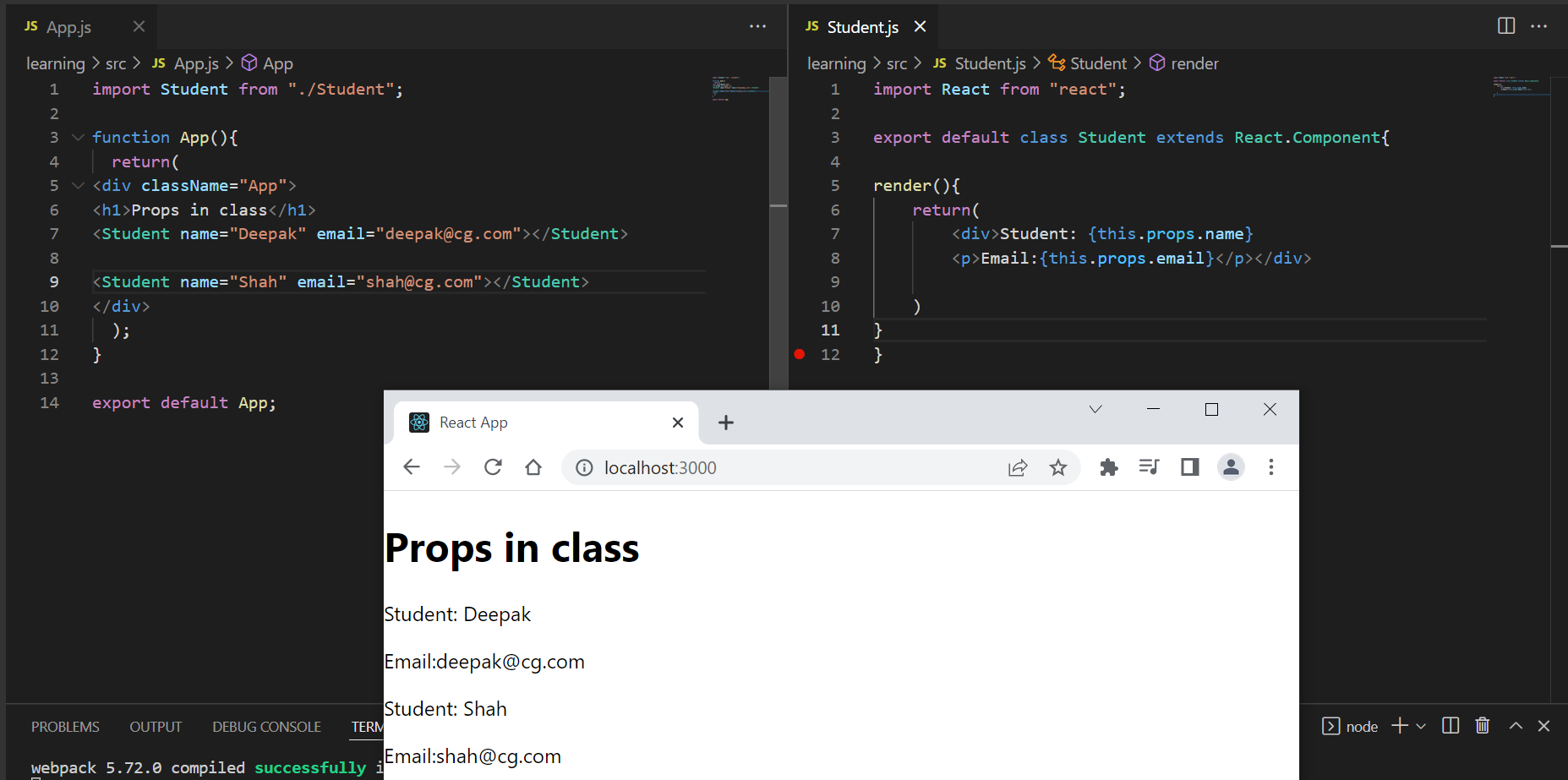
**2.b React Props (Functional and Class)**

Props stand for "**Properties**." They are **read-only** components. It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.

Props are **immutable** so we cannot modify the props from inside the component. Inside the components, we can add attributes called props. These attributes are available in the component as **this.props** and can be used to render dynamic data in our render method.

**If you want to pass** **data(data transfer) from one component to another we use props(like parameters functions)**

1. Props with functional component   
   
2. Props with class



1. Pass function as props

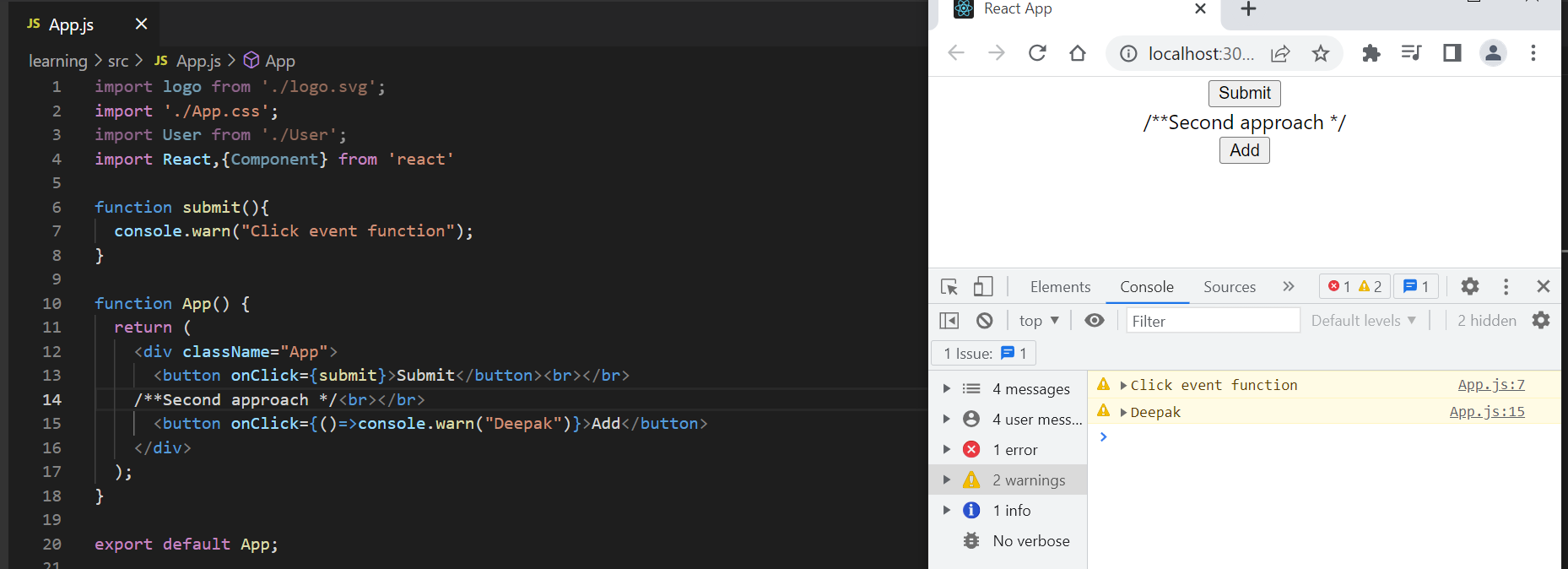


1. Event Handling

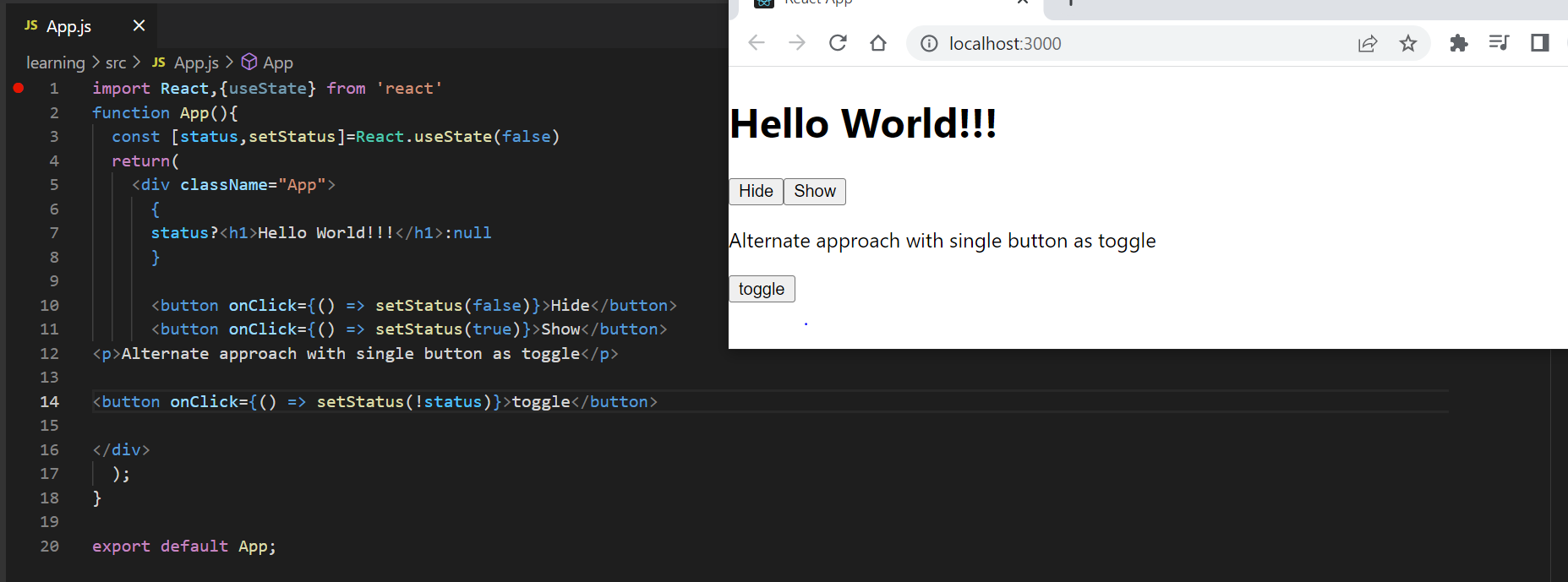
Just like HTML DOM events, React can perform actions based on user events.

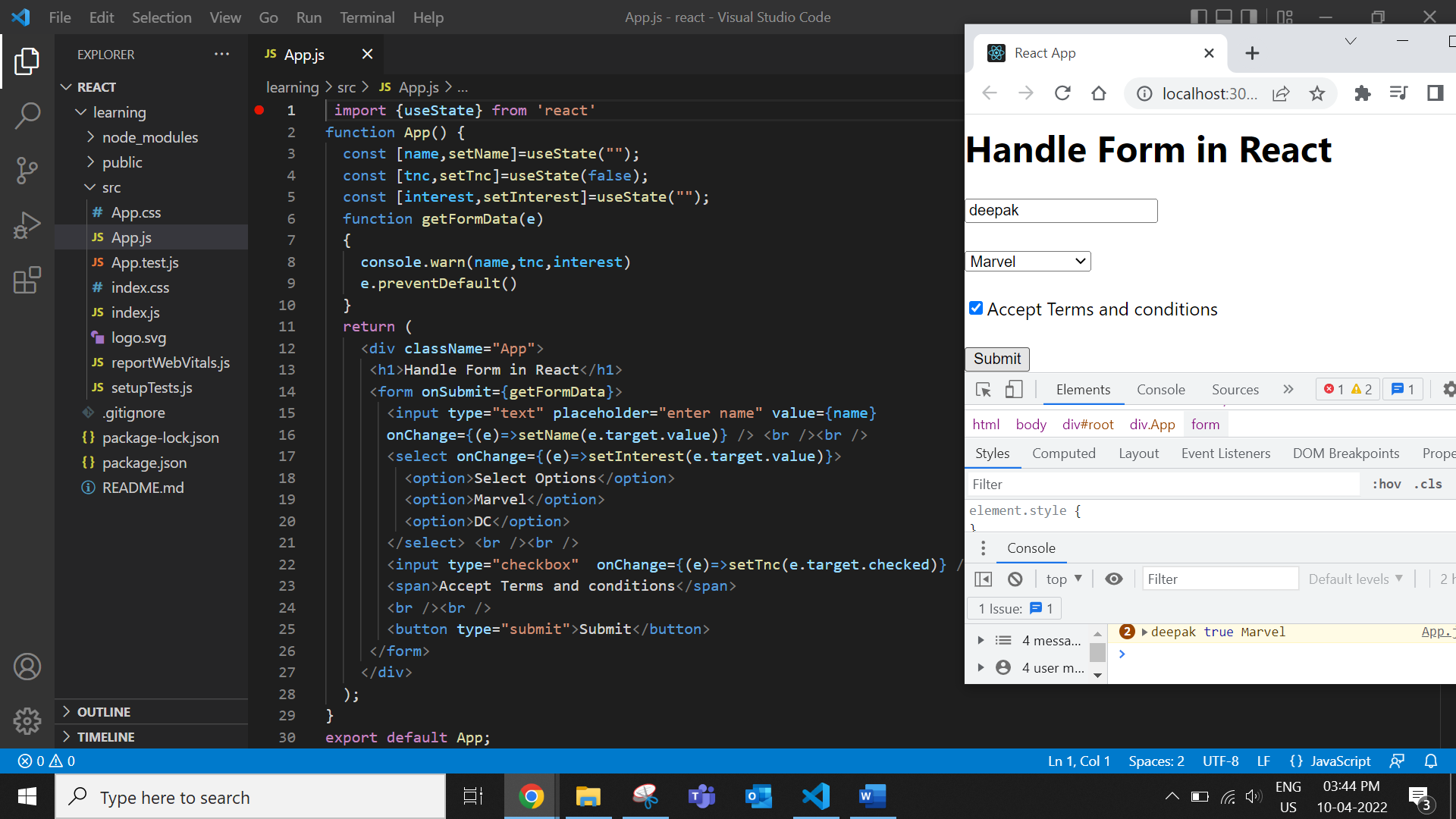
React has the same events as HTML: click, change, mouseover etc.

1. On Click event

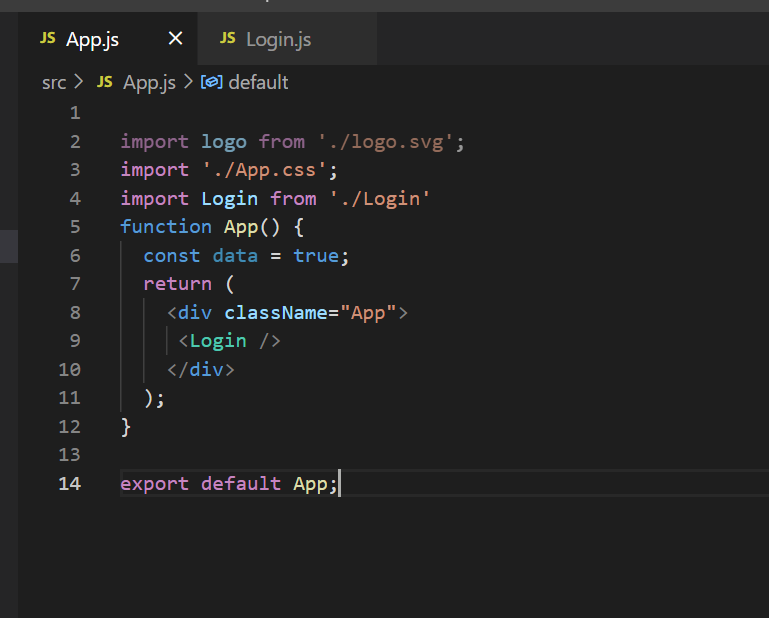


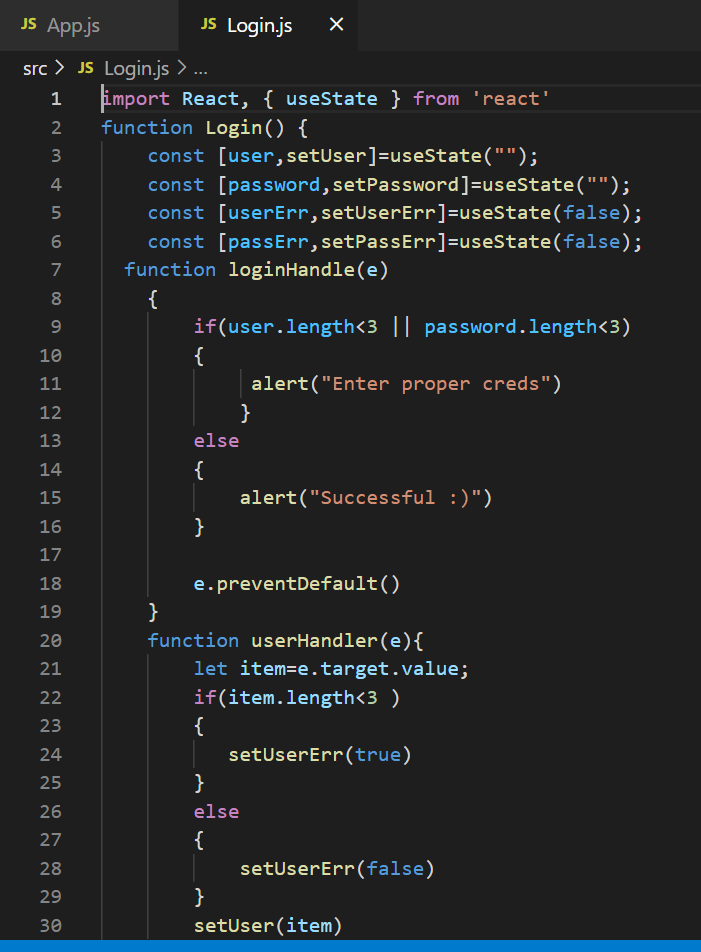
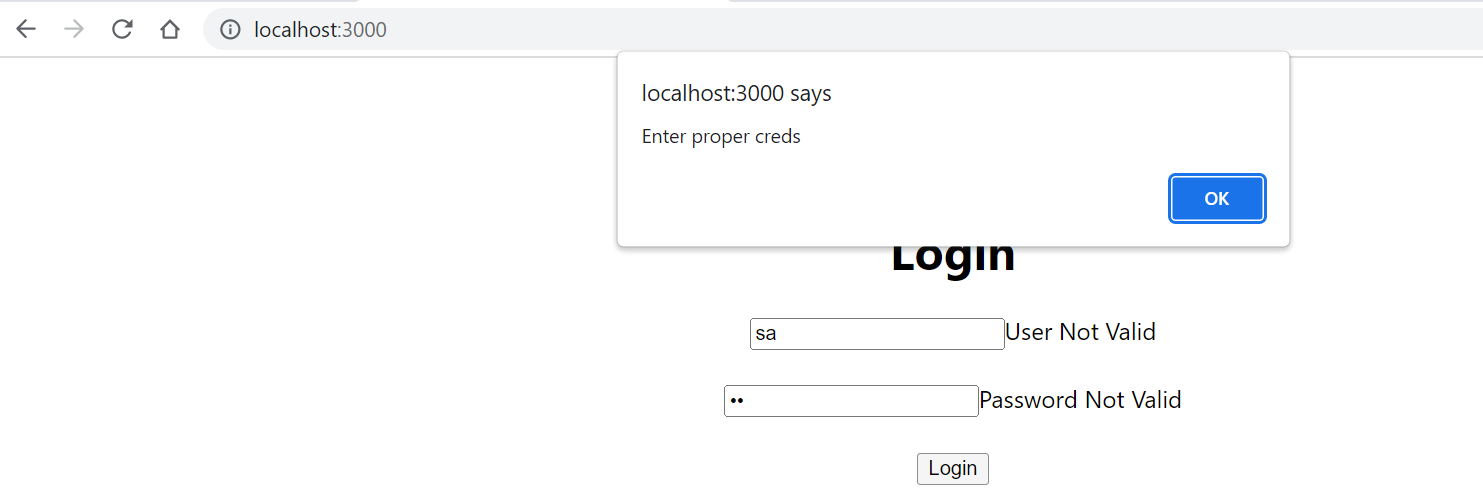
Hide, Show and Toggle



Form Handling   


Basic validation in react :



1. Life Cycle of React

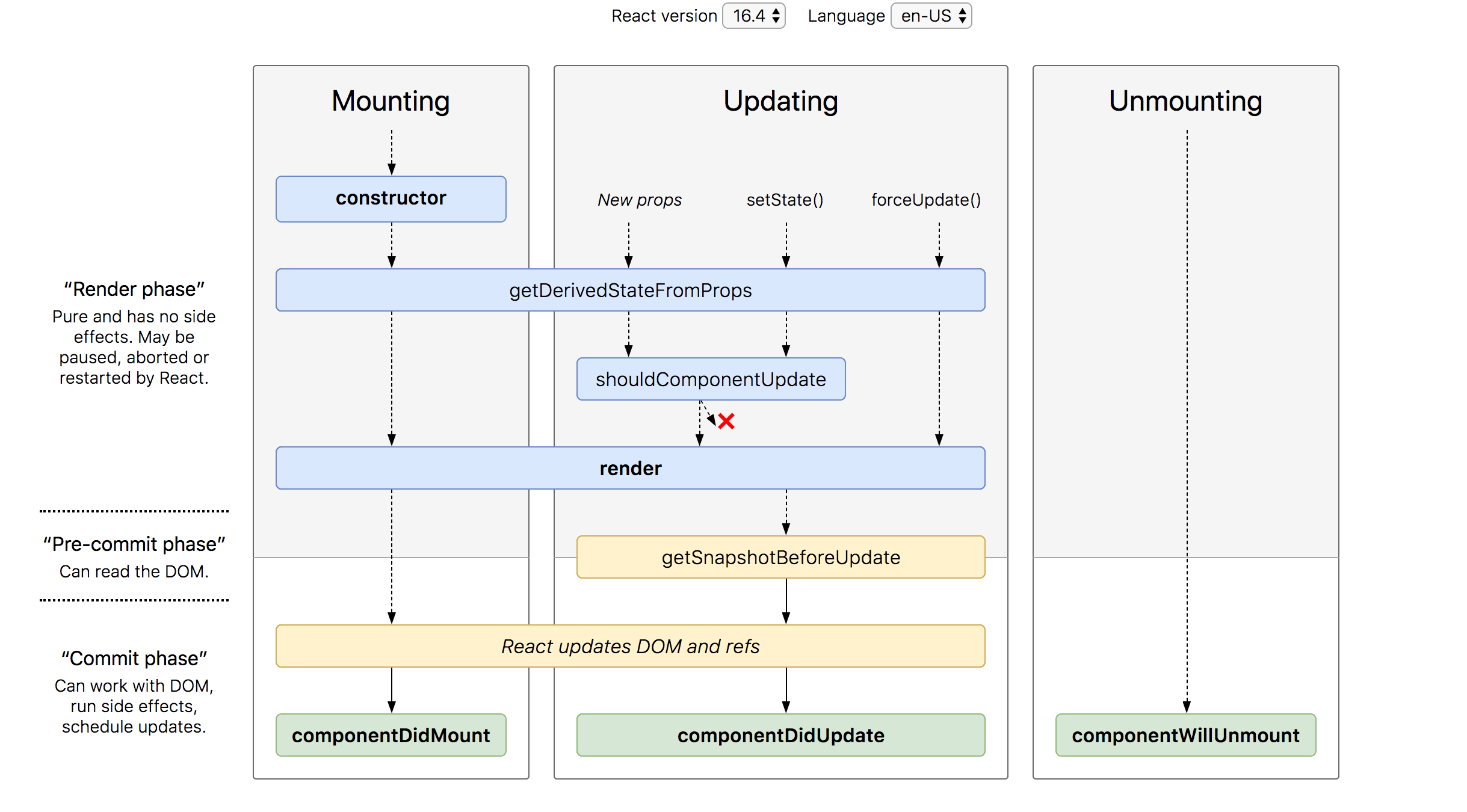
In React, Life cycle of a component represents the different stages of the component during its existence. React provides callback function to attach functionality in each and every stages of the React life cycle.

**Life cycle API**

Each React component has three distinct stages.

* **Mounting** − Mounting represents the rendering of the React component in the given DOM node.
* **Updating** − Updating represents the re-rendering of the React component in the given DOM node during state changes / updates.
* **Unmounting** − Unmounting represents the removal of the React component.

React provides a collection of life cycle events (or callback API) to attach functionality, which will to be executed during the various stages of the component. The visualization of life cycle and the sequence in which the life cycle events (APIs) are invoked as shown below.

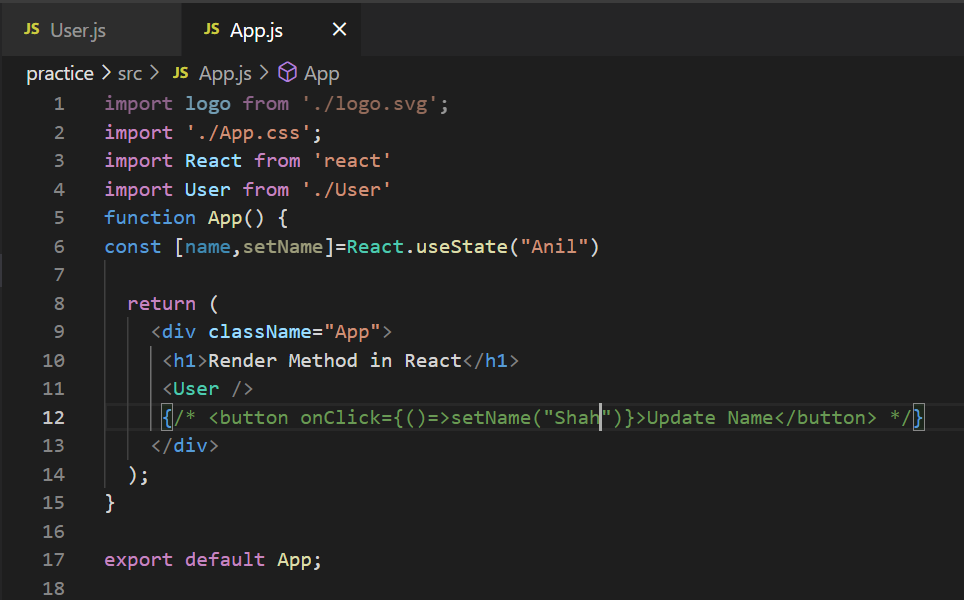


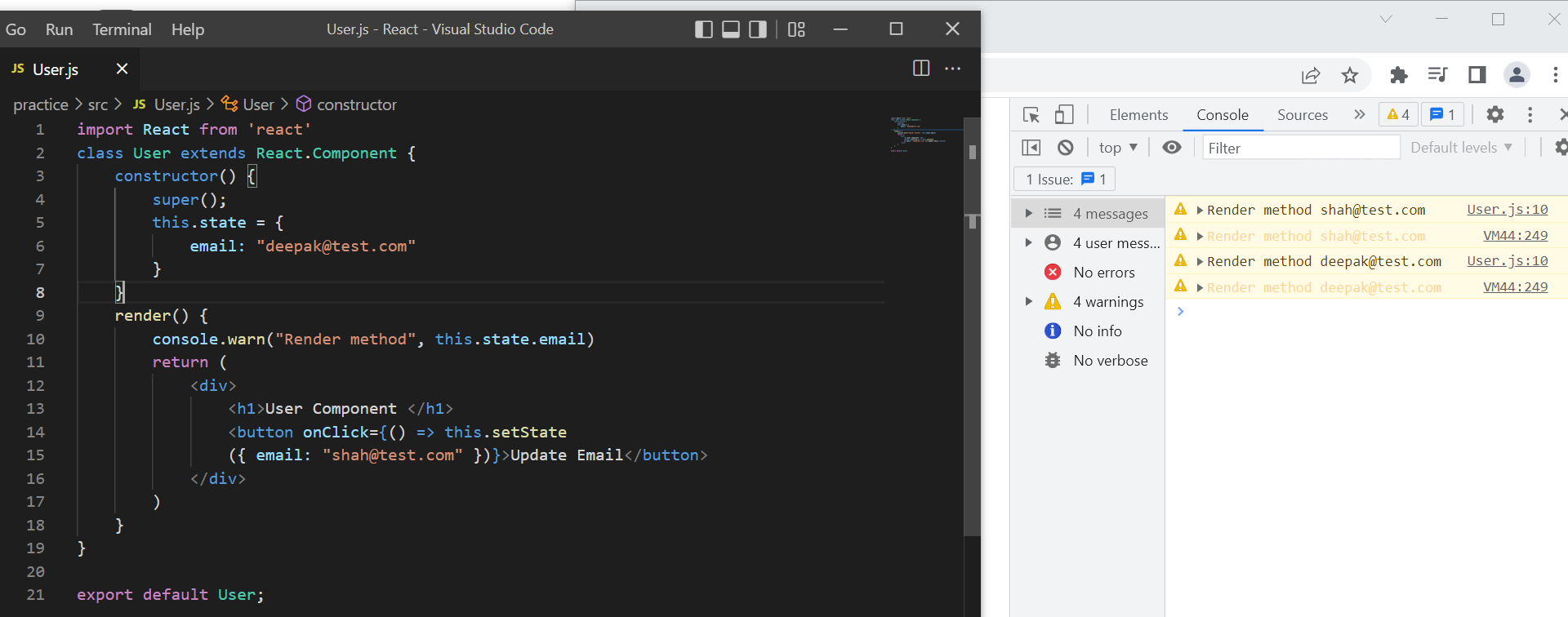
**constructor*()*** − Invoked during the initial construction phase of the React component. Used to set initial state and properties of the component.

Graphical user interface, text

Description automatically generated

**render*()*** − Invoked after the construction of the component is completed. It renders the component in the virtual DOM instance. This is specified as mounting of the component in the DOM tree.



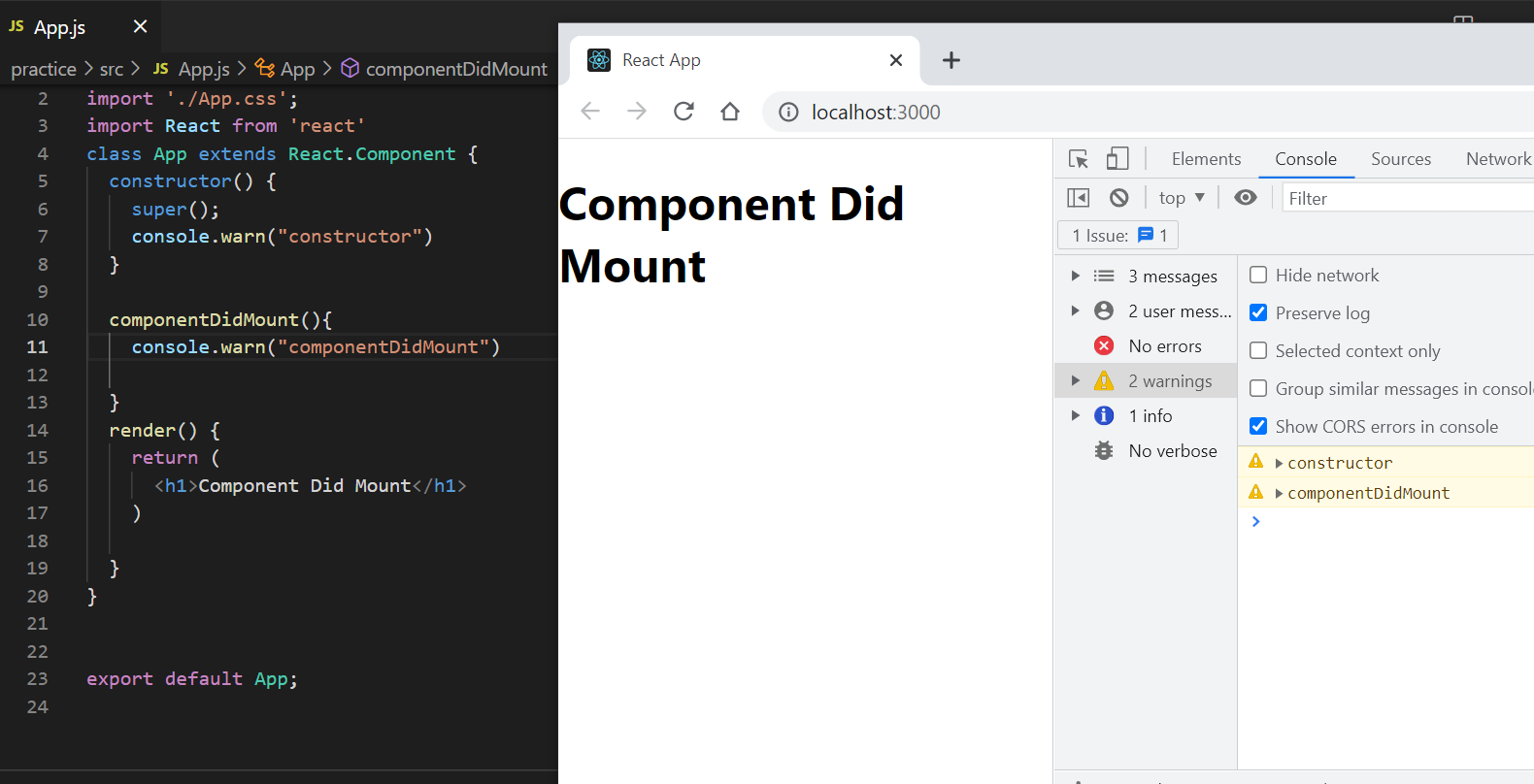


**componentDidMount*()*** − Invoked after the initial mounting of the component in the DOM tree. It is the good place to call API endpoints and to do network requests. In our clock component, setInterval function can be set here to update the state (current date and time) for every second.

componentDidMount() {

this.timeFn = setInterval( () => this.setTime(), 1000);

}



**componentDidUpdate*()*** − Similar to ComponentDidMount() but invoked during the update phase. Network request can be done during this phase but only when there is difference in component’s current and previous properties.

The signature of the API is as follows −

componentDidUpdate(prevProps, prevState, snapshot)

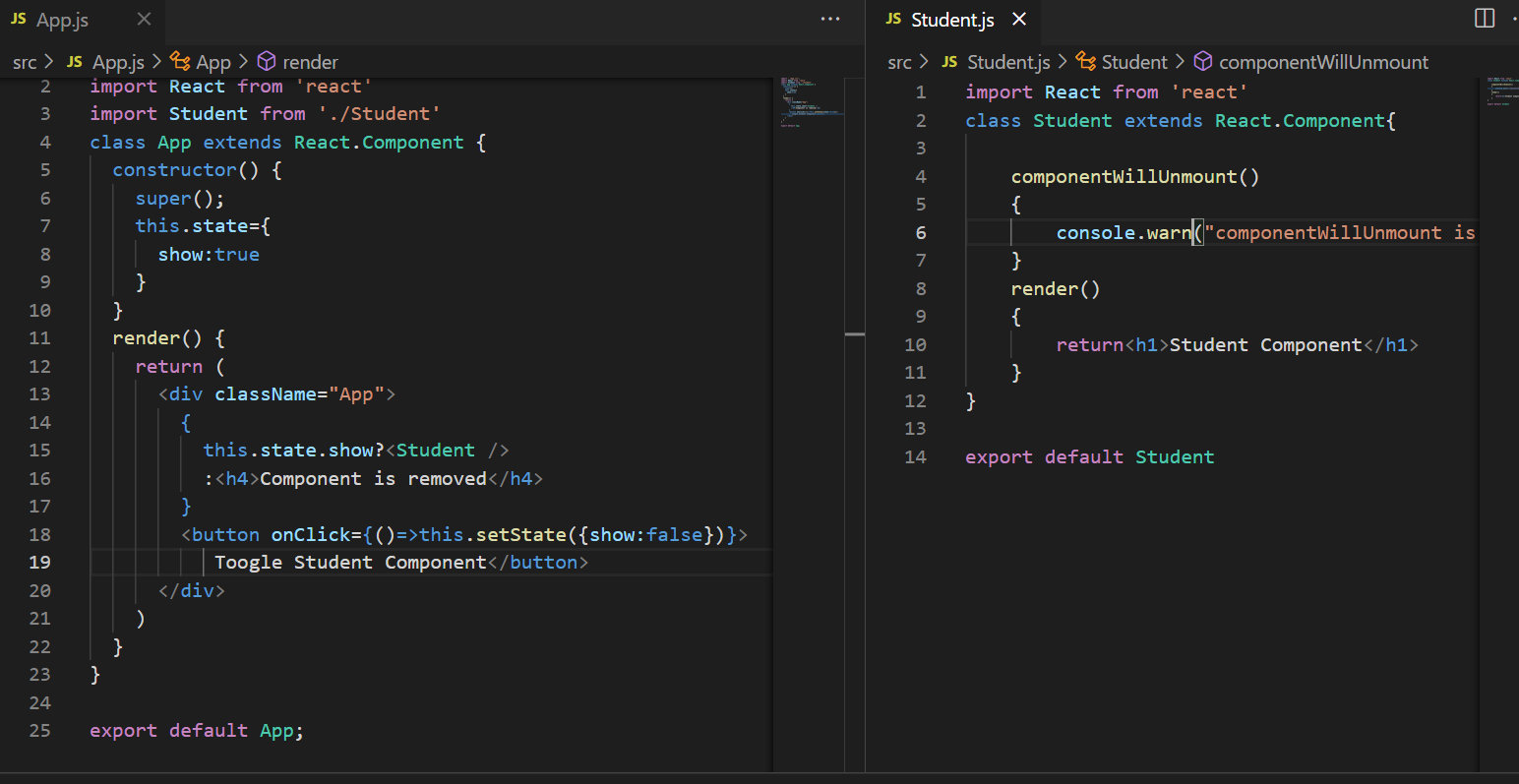
* **prevProps** − Previous properties of the component.
* **prevState** − Previous state of the component.
* **snapshot** − Current rendered content.

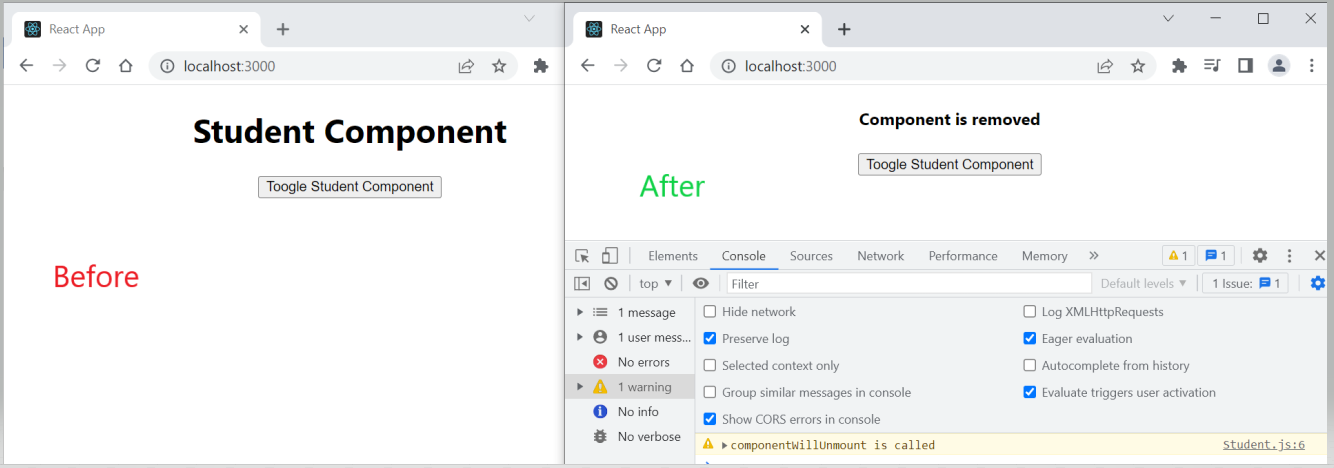
**componentWillUnmount*()*** − Invoked after the component is unmounted from the DOM. This is the good place to clean up the object. In our clock example, we can stop updating the date and time in this phase.

componentDidMount() {

this.timeFn = setInterval( () => this.setTime(), 1000);

}



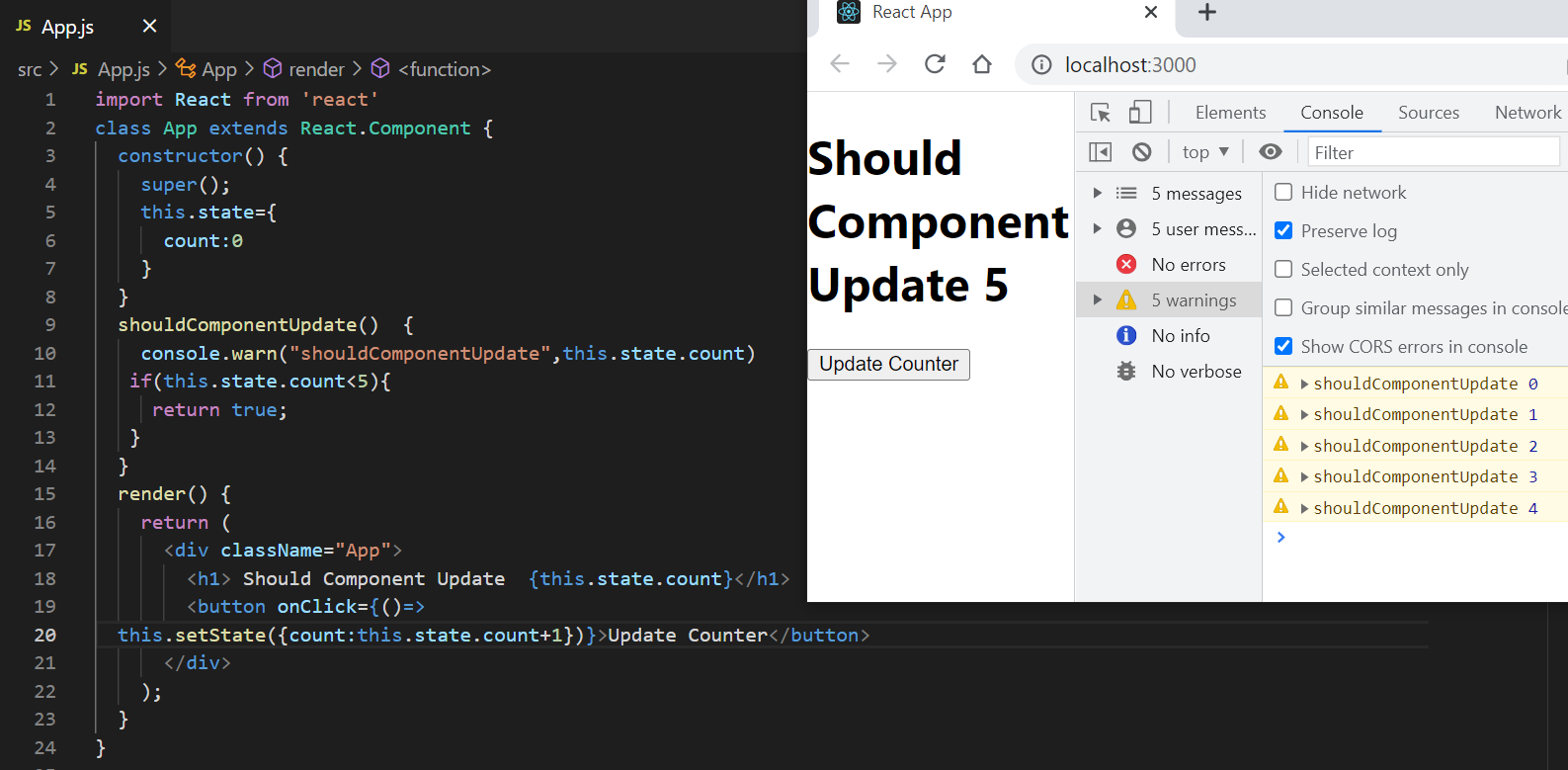


**shouldComponentUpdate*()*** − Invoked during the update phase. Used to specify whether the component should update or not. If it returns false, then the update will not happen.

The signature is as follows −

shouldComponentUpdate(nextProps, nextState)

* **nextProps** − Upcoming properties of the component
* **nextState** − Upcoming state of the component



**getDerivedStateFromProps** − Invoked during both initial and update phase and just before the render() method. It returns the new state object. It is rarely used where the changes in properties results in state change. It is mostly used in animation context where the various state of the component is needed to do smooth animation.

The signature of the API is as follows −

static getDerivedStateFromProps(props, state)

* *props* − current properties of the component
* *state* − Current state of the component

This is a static method and does not have access to this object.

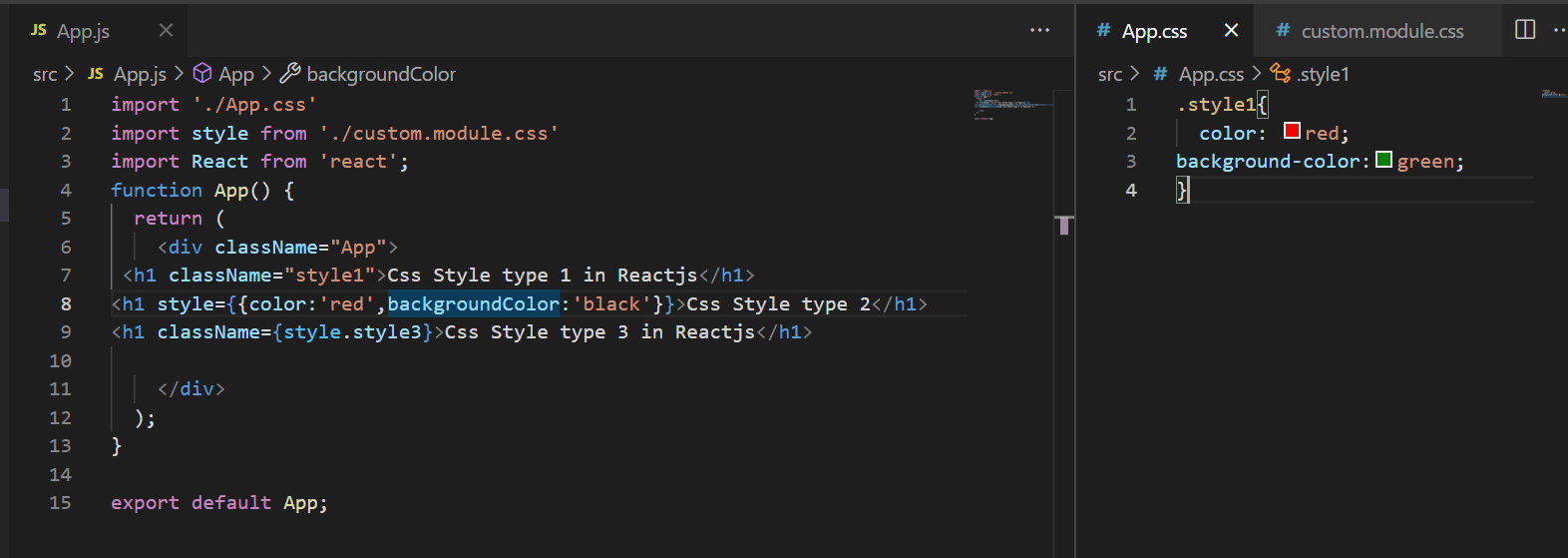
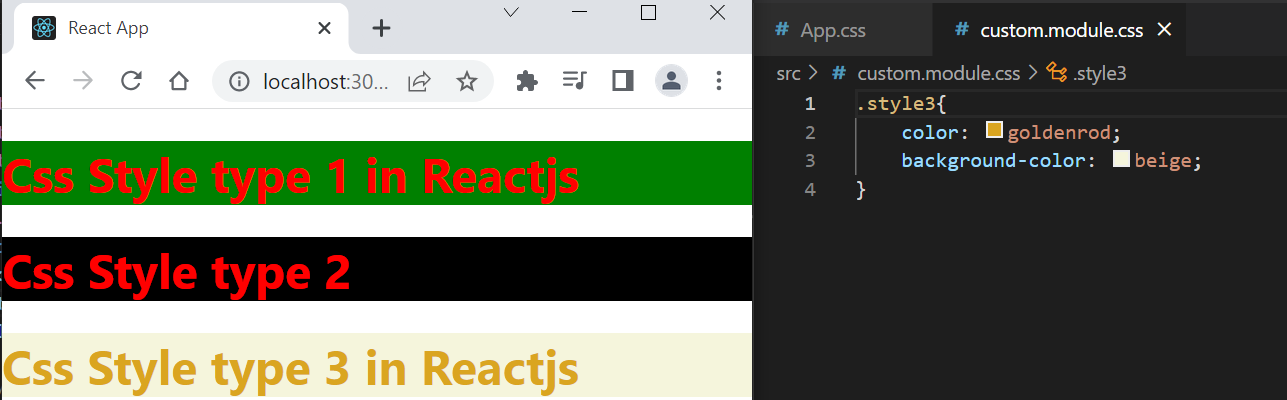
**getSnapshotBeforeUpdate** − Invoked just before the rendered content is commited to DOM tree. It is mainly used to get some information about the new content. The data returned by this method will be passed to ComponentDidUpdate() method. For example, it is used to maintain the user’s scroll position in the newly generated content. It returns user’s scroll position. This scroll position is used by componentDidUpdate() to set the scroll position of the output in the actual DOM.

The signature of the API is as follows −

getSnapshotBeforeUpdate(prevProps, prevState)

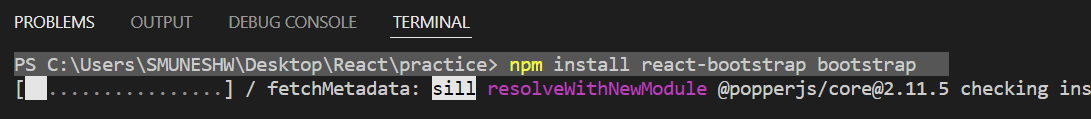
* prevProps − Previous properties of the component.
* prevState − Previous state of the component.

1. CSS in react js

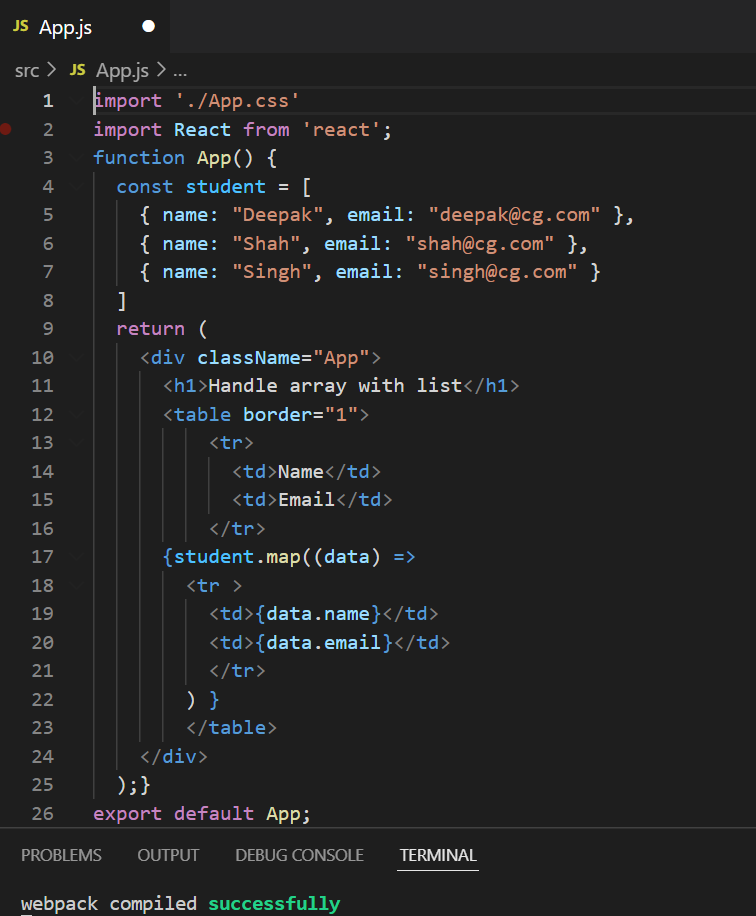
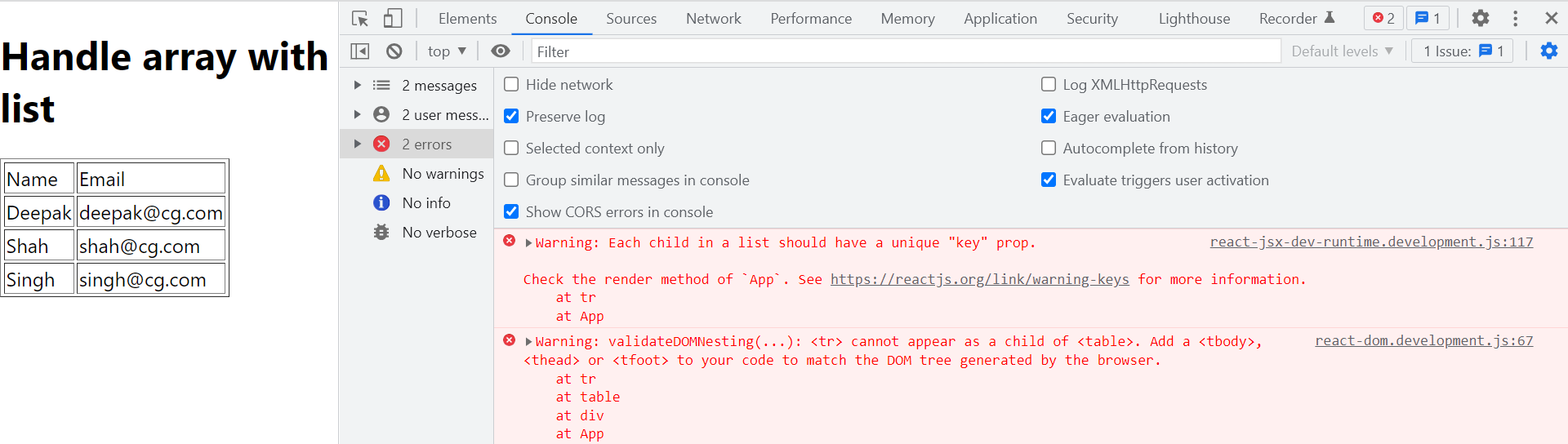
  


Install Bootstrap:

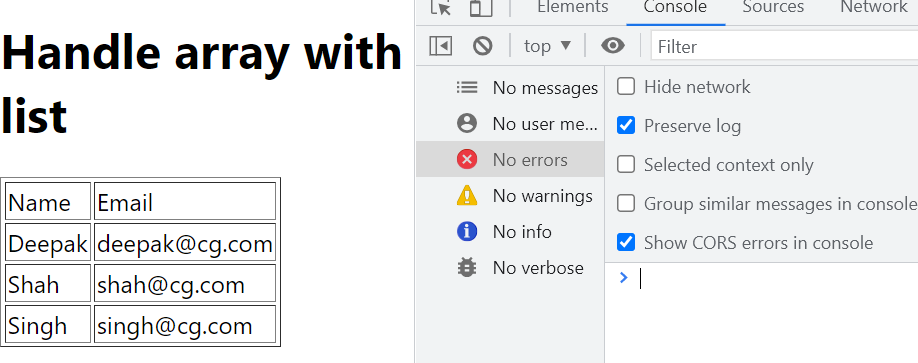
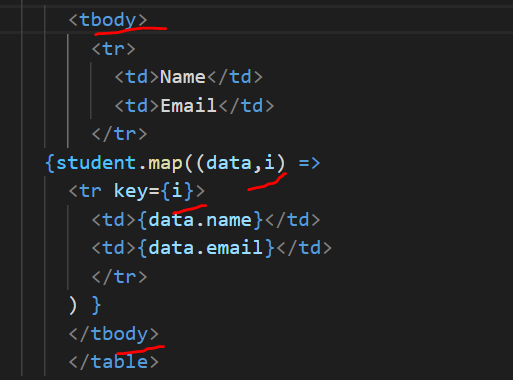
npm install react-bootstrap bootstrap



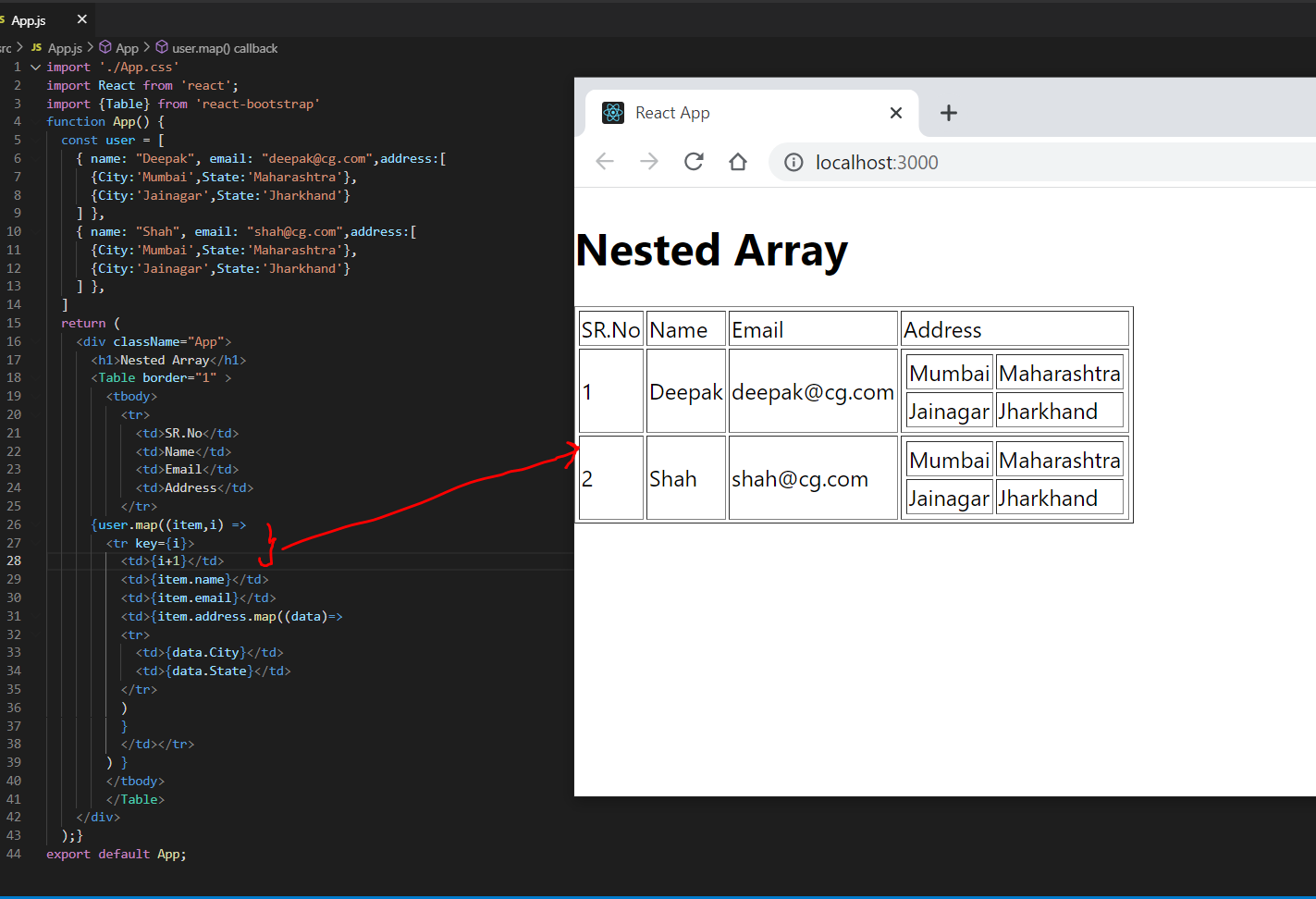
Learn more from https://react-bootstrap.github.io/

1. Handle array with list  
    

We are getting these two warnings at console.



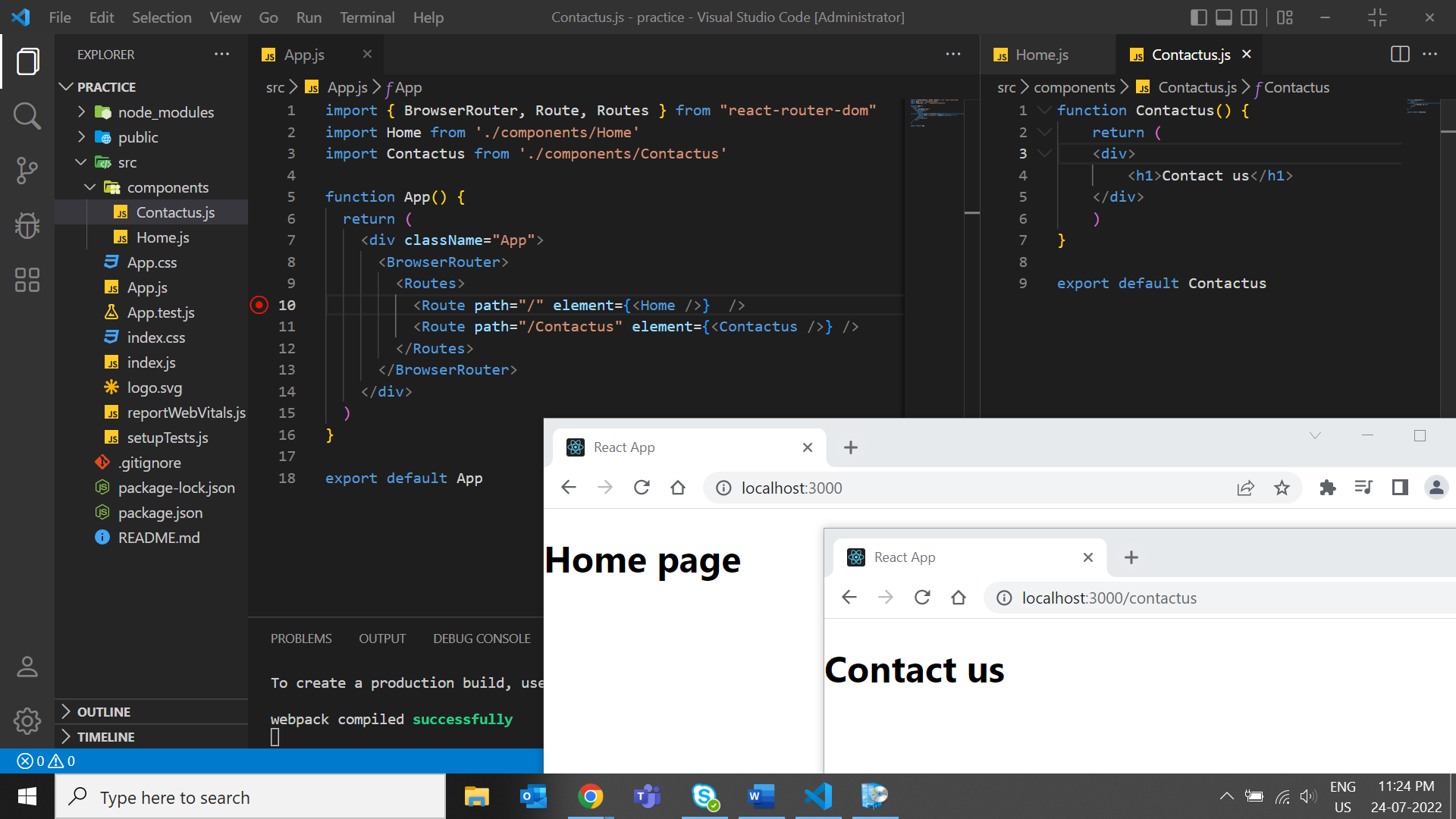
Solution of the issue

Nested Array:  


1. React Router

To install react router cmd

* npm install react-router-dom

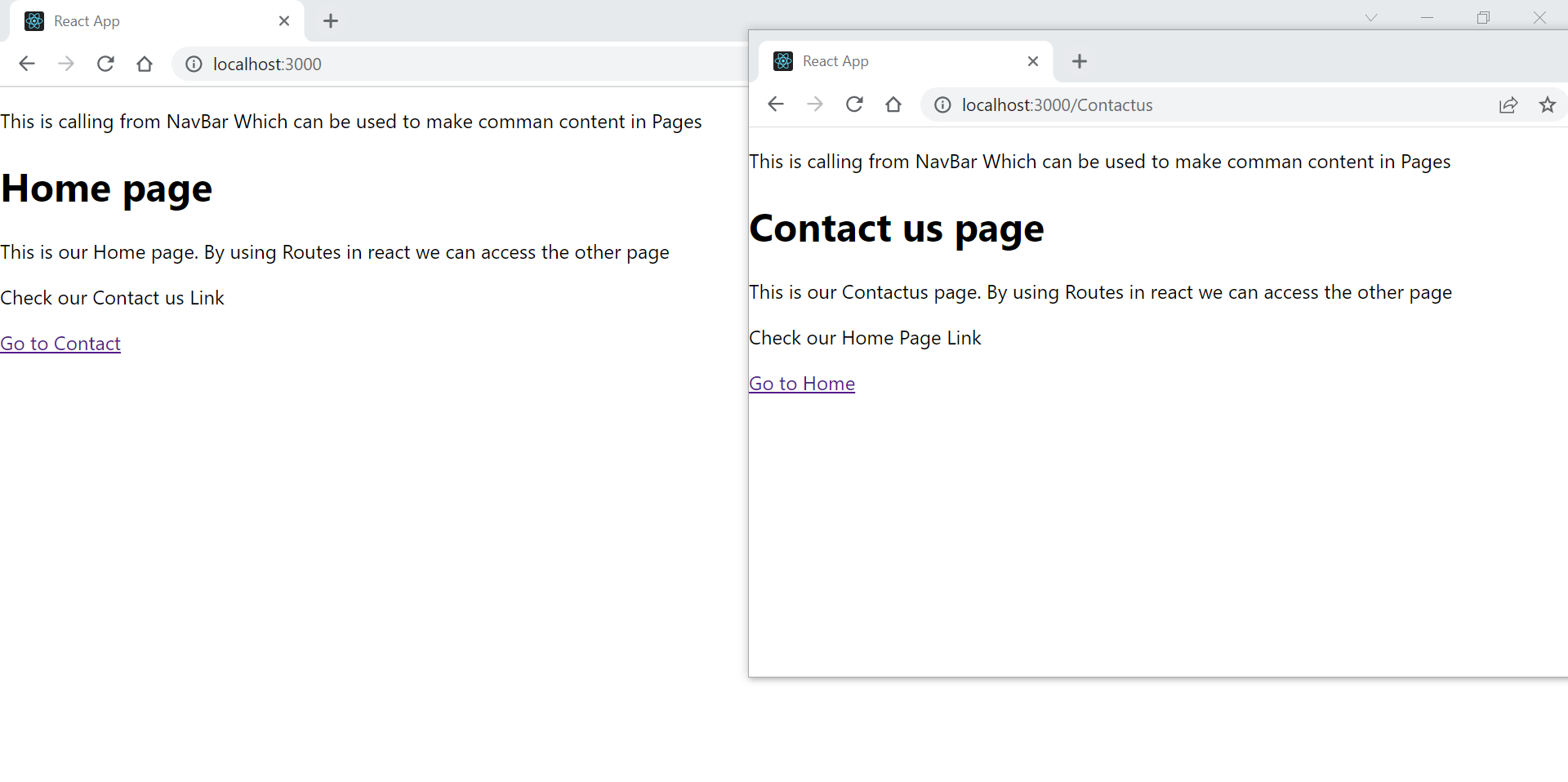


Same code for home.js as contactus

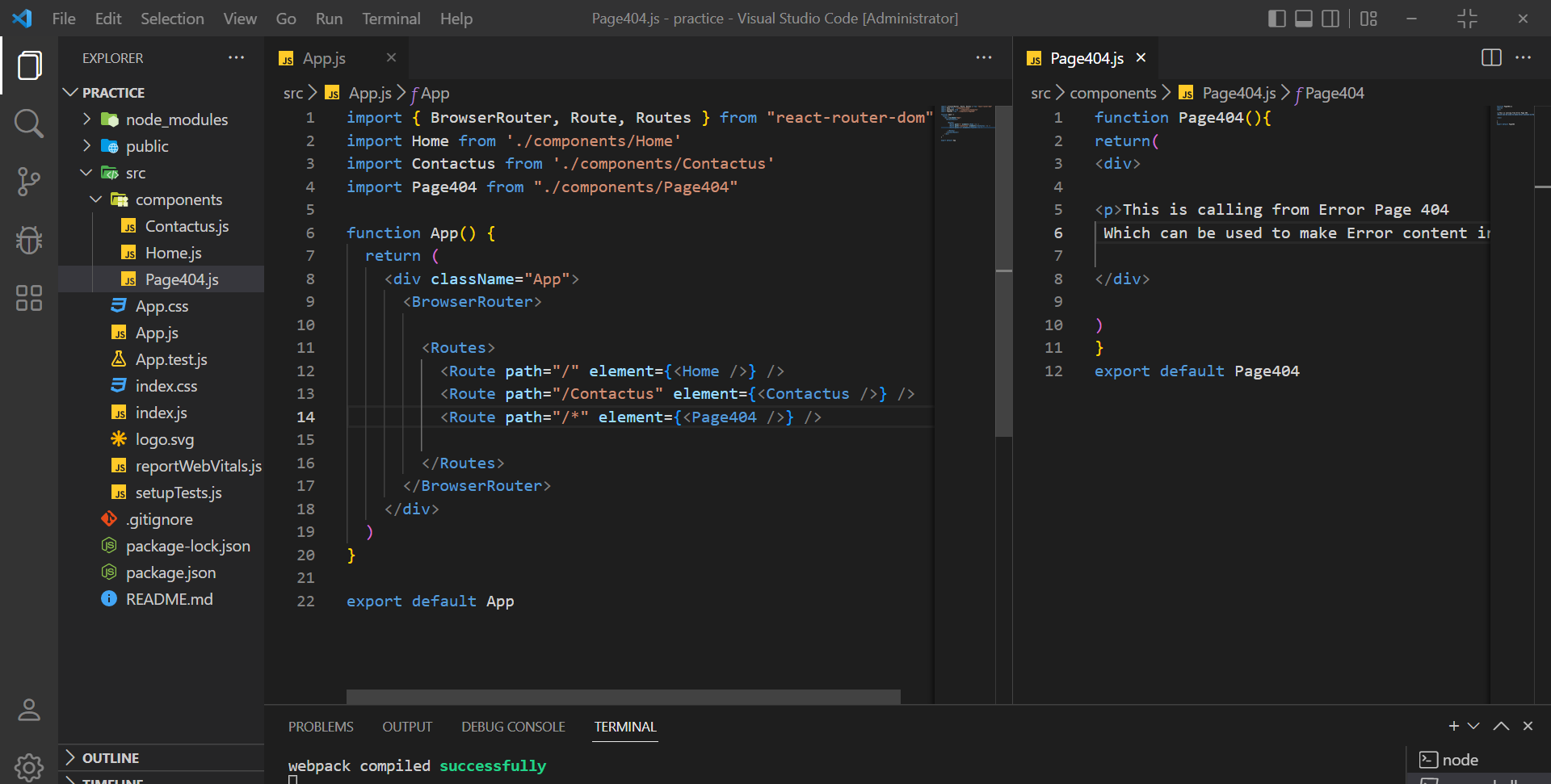
**NavBar**

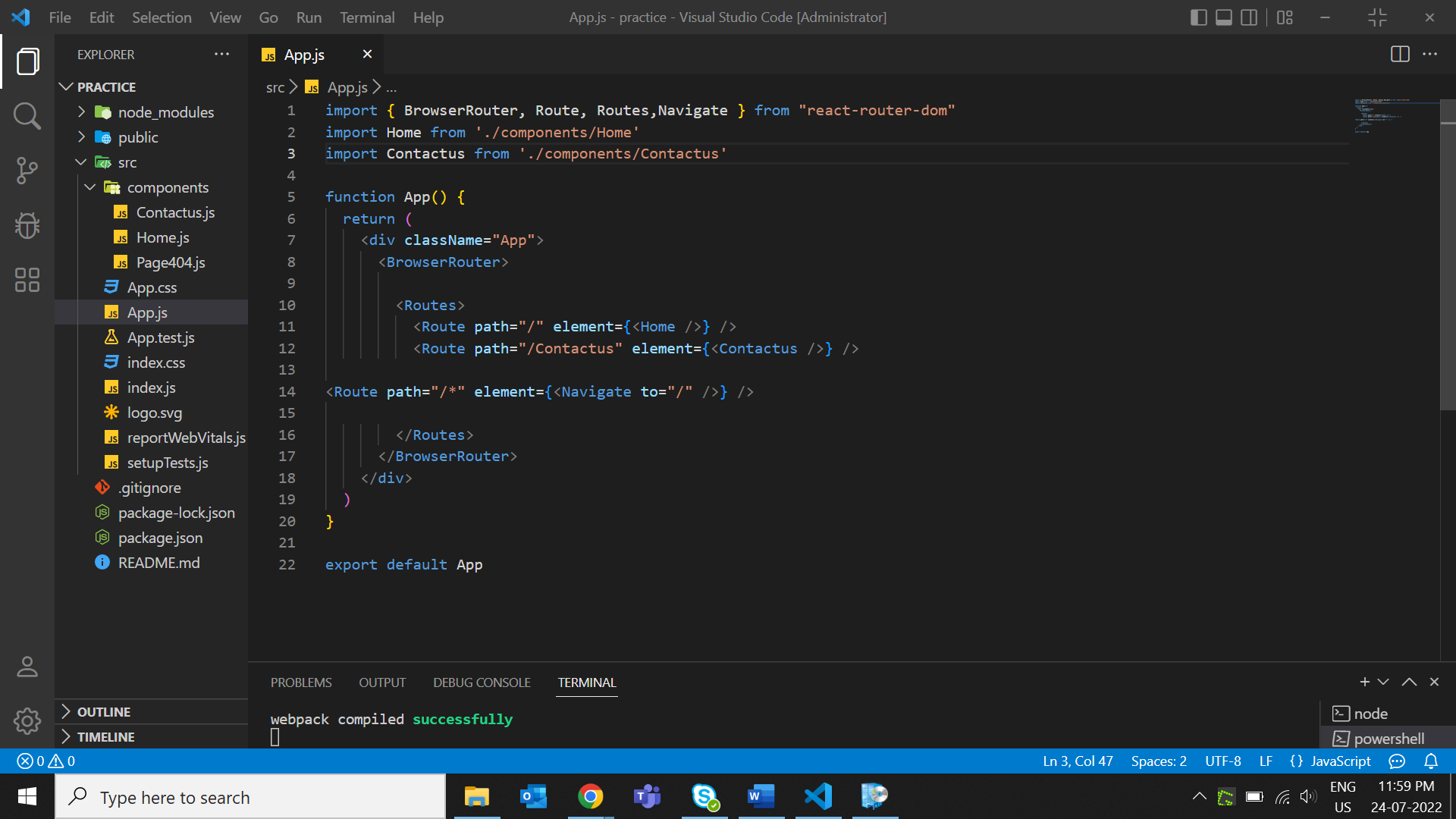






Error Page404



When u want to navigate user to home page if he is changing your url to any other and you want to navigate user to home page again  


1. Promise :

*"I Promise a Result!"*

"Producing code" is code that can take some time

"Consuming code" is code that must wait for the result

A Promise is a JavaScript object that links producing code and consuming code

Promise Object Properties

A JavaScript Promise object can be:

* Pending
* Fulfilled
* Rejected

The Promise object supports two properties: **state** and **result**.

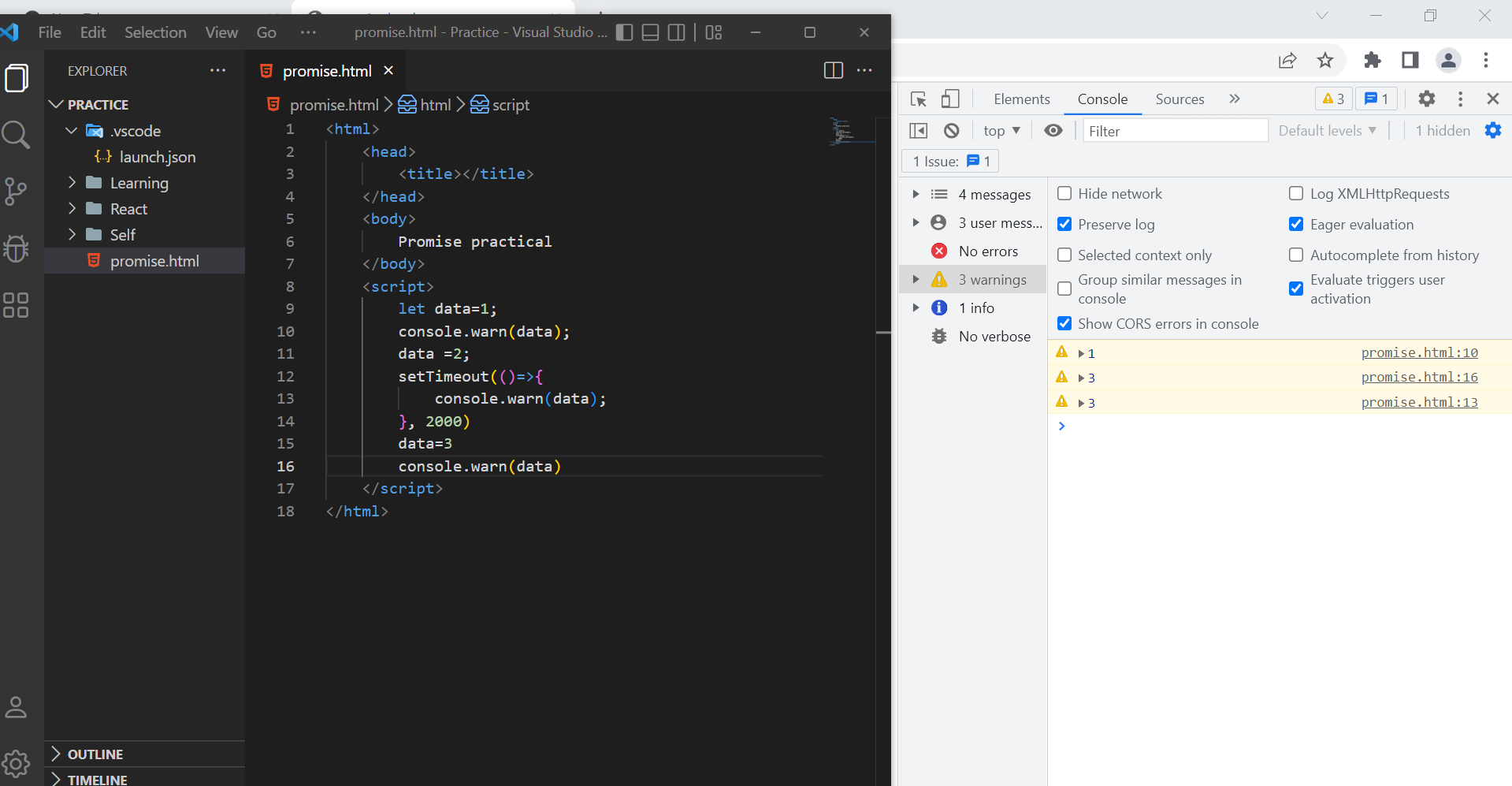
While a Promise object is "pending" (working), the result is undefined.

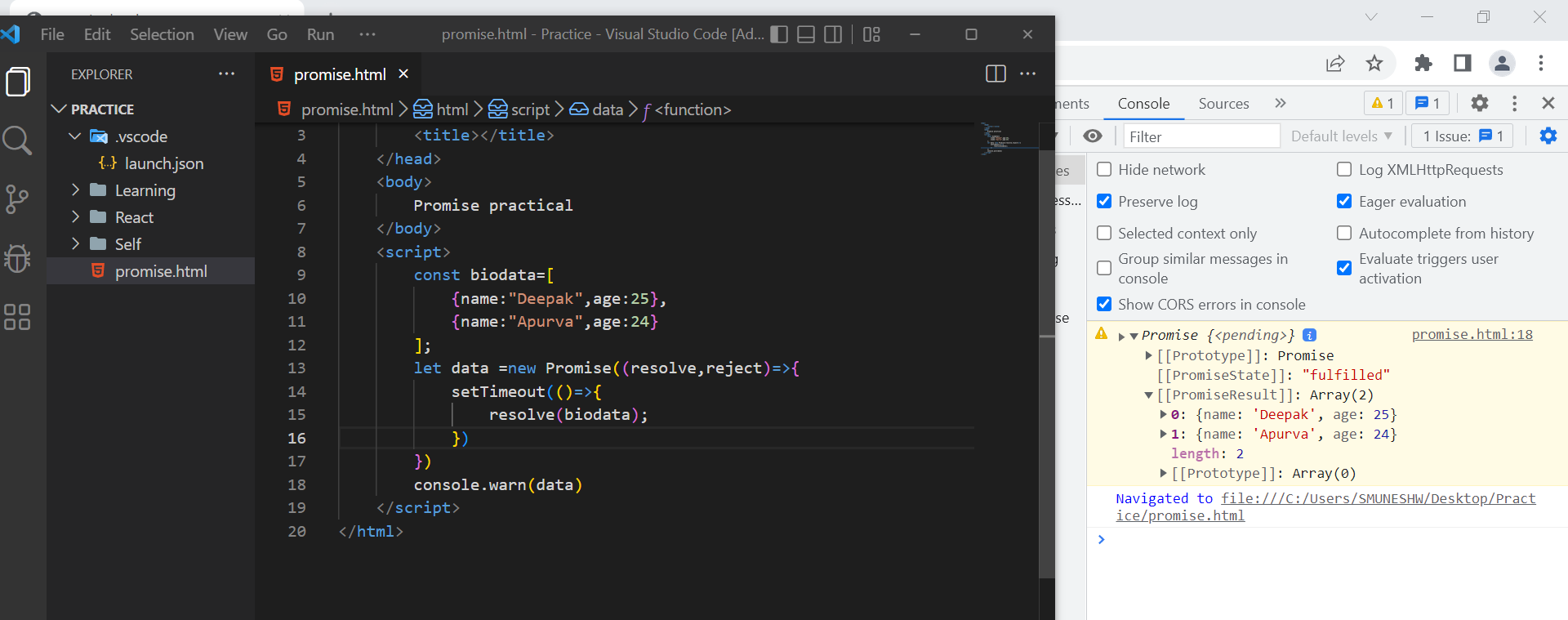
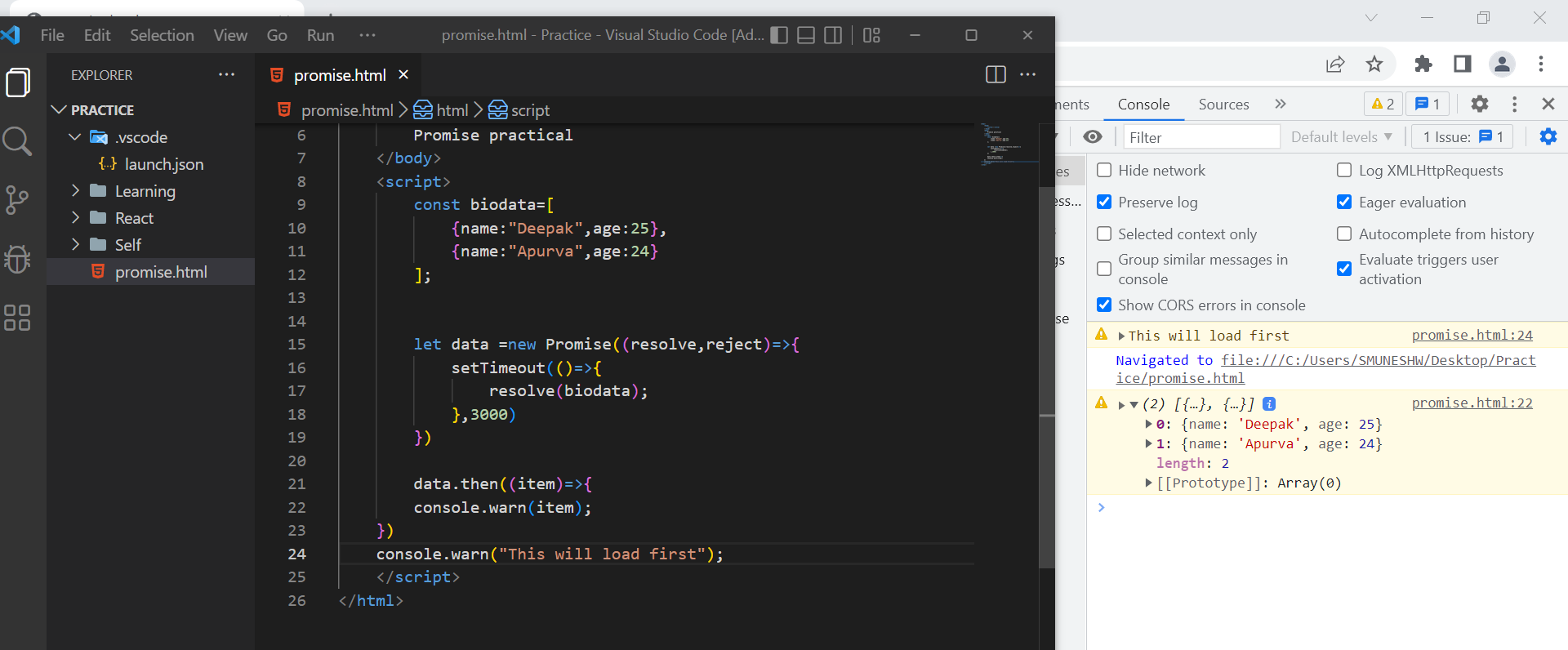
When a Promise object is "fulfilled", the result is a value.

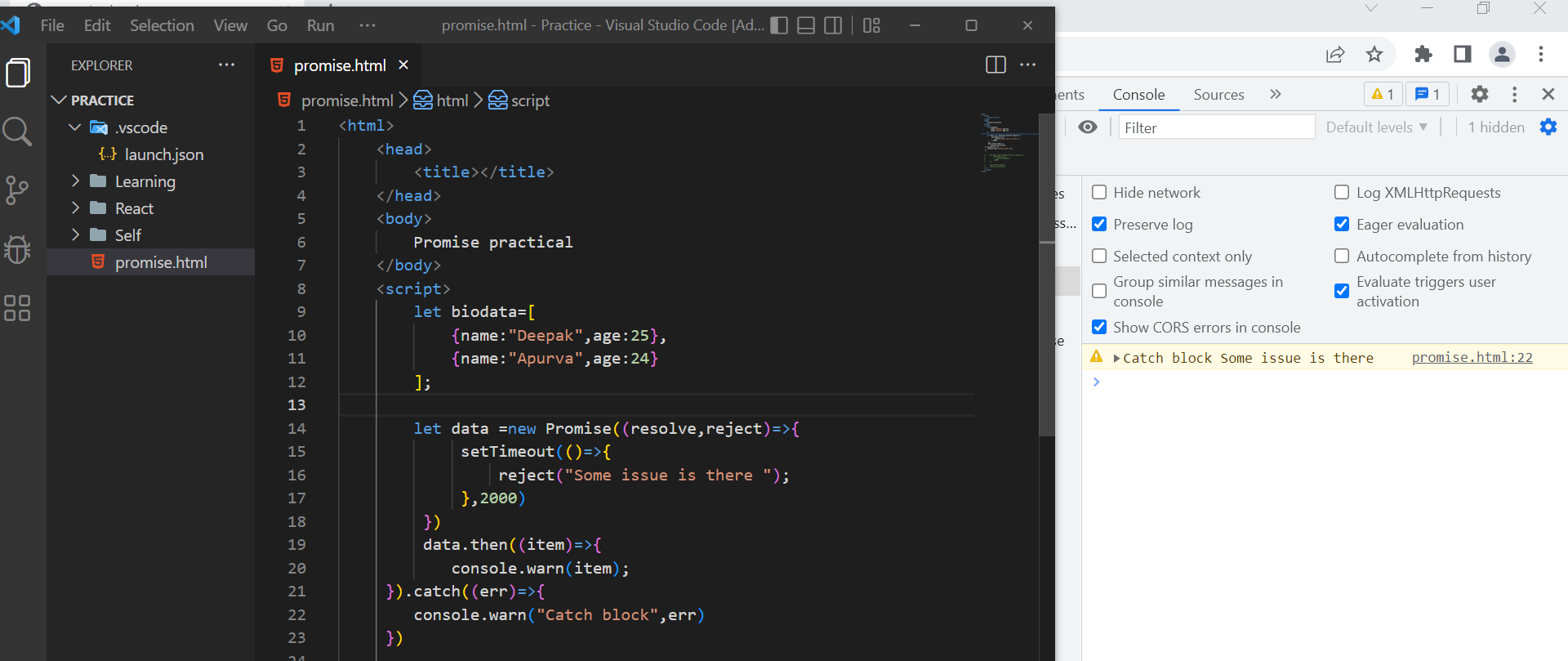
When a Promise object is "rejected", the result is an error object.

The problem with the javascript as when you do a settimeout then data lose so to resolve such issues. Promise came into picture.

Promise and Asyn await is a Javascript ECMA 8 concept not react.



  
But this is showing data in the form of inside an promise not the exact data in order to get that you use  
 then keyword  




API Calls

Get API Method:

