**Instructions**

Close

**Lab instructions**

1 of 1 completed

Managing state in React

Success: Complete

**Complete**

​

**Task**

In the starter code of this code lab, you are given a **Fruits** component that has its own state. Based on this state, it outputs three fruits on the screen. Additionally, you have a **FruitsCounter** component which shows a message that reads: "Total fruits: 2".

Your task is to lift state up from the **Fruits** component to the **App** component, so that you can then pass the state information to both the **Fruits** component and the **FruitsCounter** component.

This change to the app should fix the previously incorrect message of "Total fruits: 2". The new message should be "Total fruits: 3". However, the new message will not be just a hard-coded string. Instead, it should reflect the number of fruits that exist in the state variable, so based on how many fruits there are in the state array, this information should affect the output of the total number of fruits - as returned from the **FruitsCounter** component.



**Steps**

**Step 1.** This task’s starting point is the stateless **App** component’s code:

3

4

5

6

7

8

9

10

11

12

13

14

15

1

2

function App() {

  return (

    <div className="App">

      <h1>Where should the state go?</h1>

      <Fruits />

      <FruitsCounter />

    </div>

  );

};

export default App;

import Fruits from "./Fruits";

import FruitsCounter from "./FruitsCounter";

The first step of this task is to move the state from the **Fruits** component to **App** component.

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

1

2

function Fruits() {

    const [fruits] = React.useState([

        {fruitName: 'apple', id: 1},

        {fruitName: 'apple', id: 2},

        {fruitName: 'plum', id: 3},

    ]);

    return (

        <div>

            {fruits.map(f => <p key={f.id}>{f.fruitName}</p>)}

        </div>

    );

};

export default Fruits;

import React from "react";

Update the App component to pass this **fruits** state as props to both the **Fruits** and **FruitsCounter** components.

**Step 2.**

Once you've moved the state up from the Fruits component to the App component,Remove the state from the  **Fruits**  component and update it to receive the  fruits  as prop and display the list of fruits dynamically based on the passed prop.

3

4

5

6

7

8

9

10

11

12

13

14

1

2

function Fruits({ fruits }) {

  return (

    <div>

      {fruits.map((f) => (

        <p key={f.id}>{f.fruitName}</p>

      ))}

    </div>

  );

}

export default Fruits;

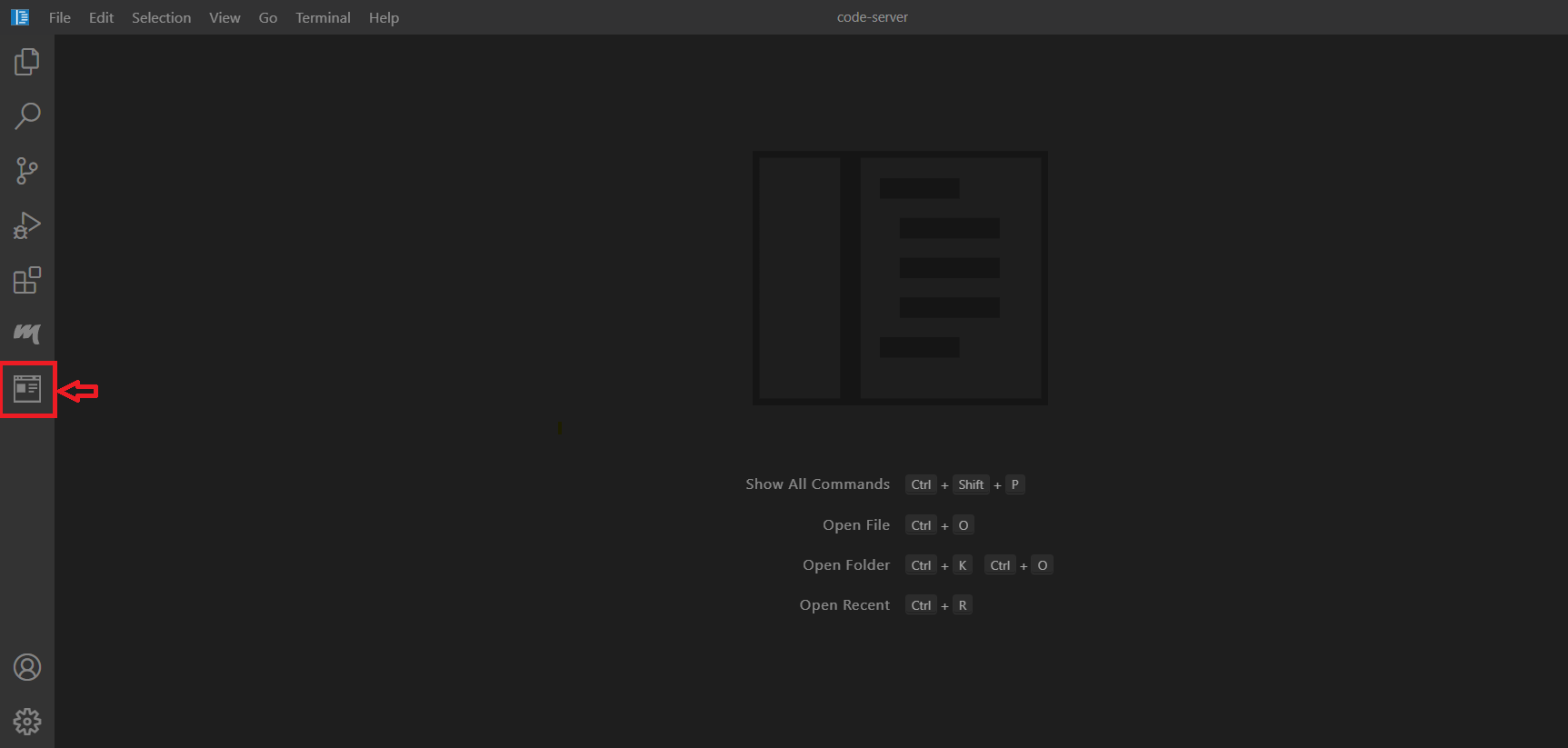
import React from "react";

**Step3.**

Now, Modify the FruitsCounter component to accept the fruits prop and calculate the total fruits dynamically using the **length** property of the array.

**Step 4.**

* At the top of the lab environment, locate the Terminal menu. Click on it to open a dropdown, then select New Terminal.Use the **npm start** command to start the development server.
* If you encounter errors like *File not found* or *Unexpected token*, in the terminal, stop the server with **Ctrl + C** and restart it using **npm start**.
* You can now view the App in your lab browser. To view the output, click on the **Browser Preview** icon located on the left panel. It is the last icon in the panel.



* When your lab browser has launched, enter: **http://localhost:3000** in the address bar to see the output.
* The app displays the correct list of fruits. The total fruits count dynamically reflects the length of the fruits array.

**Complete**